НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра системного програмування і спеціалізованих комп'ютерних систем

До захисту допущено

Завідувач кафедри

_____ Романкевич В.О.
(підпис) (ініціали, прізвище)

"___" червня 2020 р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Комп'ютерні системи та компоненти»

спеціальності **123 «Комп'ютерна інженерія»**

на тему: Програмні засоби ідентифікації рухомих об'єктів в відео-потоці _____
_____
_____

Виконав :

студент IV курсу, групи КВ-63
(шифр групи)

Бобров Владислав Сергійович _____ _____
(прізвище, ім'я, по батькові) (підпис)

Керівник доц.каф.СПСКС , к.т.н. Петрашенко А.В _____ _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант з нормоконтролю, доц.каф.СПСКС, к.т.н. Клятченко Я.М. _____
(назва розділу) (посада, вчене звання, науковий ступінь, прізвище, ініціали) (підпис)

Рецензент _____ _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2020 року

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра системного програмування і спеціалізованих комп‘ютерних систем

Рівень вищої освіти – перший (бакалаврський)

Спеціальність 123 «Комп’ютерна інженерія»

Освітньо-професійна програма «Комп’ютерні системи та компоненти»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Романкевич В.О.
(підпис) (ініціали, прізвище)

«___» червня 2020 р.

ЗАВДАННЯ
на дипломний проєкт студента
Боброва Владислава Сергійовича
(прізвище, ім’я, по батькові)

1. Тема проєкту Програмні засоби ідентифікації рухомих об'єктів в відео-потоці _____
_____
_____,
керівник проєкту доц.каф. СПСКС, к.т.н. Петрашенко А.В. _____,
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «29» травня 2020р. №_____

2. Термін подання студентом проєкту _____

3. Вихідні дані до проєкту див. Технічне завдання

4. Зміст пояснювальної записки
- Аналіз існуючих рішень
- Проблематика розпізнавання
- Порівняння інструментів для реалізації програми
- Архітектура розробленої системи

5. Перелік графічного матеріалу (із зазначенням обов’язкових креслеників, плакатів, презентацій тощо)

- Архітектура розробленої загорткової нейронної мережі. Схема алгоритму

- Загальна схема проекту. Схема структурна
- Блок-схема розробленої. Схема алгоритму
- Схема виявлення руху. Схема структурна

6. Консультанти розділів проєкту[1*]

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
нормоконтроль	к. т. н., доцент Клятченко Я.М.		

7. Дата видачі завдання _____

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1	Вивчення літератури за тематикою проекту	10.03.2020	
2	Розроблення та узгодження технічного завдання	28.03.2020	
3	Аналіз існуючих рішень	11.04.2020	
4	Підготовка матеріалів розділів дипломного проекту	15.04.2020	
5	Підготовка звіту дипломного проекту	10.05.2020	
6	Передзахист дипломного проекту	20.05.2020	

Студент _____ (підпис) _____ (ініціали, прізвище)

Керівник проєкту _____ (підпис) _____ (ініціали, прізвище)

[1] * Консультантом не може бути зазначено керівника дипломного проекту.

АННОТАЦІЯ

Мета дипломного проєкту дослідити теоретичні відомості про засоби ідентифікації та знаходження об’єктів, а також, створити на соснові досліджень власну програму, здатну знаходити та ідентифікувати об’єкти.

Для дослідження теми проєкту було реалізовано різні методи виявлення об’єктів на зображенні й алгоритми їх обробки, для покращення швидкості і точності.

В результаті було створено програму, здатну з великою точністю оброблювати відео-потоки різної якості, класифікувати об’єкти на зображенні та показувати їх місцезнаходження. Перевагою розробленої програми є здатність оброблювати саме рухомі об’єкти в відео-потоці, що зменшує навантаження на процесор і пришвидшує роботу системи. Також були проведені дослідження розробленої системи і продемонстровано роботу програми на прикладі системи відеоспостереження.

Дипломний проєкт містить: 64 ст., 32 рис., 2 табл., 9 посилань на використаних джерел.

Ключові слова: ідентифікація об’єктів, обробка зображень, комп’ютерний зір, OpenCV, CNN.

SUMMARY

The purpose of the diploma project is to explore the theoretical information about methods and algorithms in indification and finding objects and on basis of the acquired knowledge to develop your own program capable of finding and identifying objects.

For research of the project theme different methods of object detection in the image and algorithms of their processing were implemented to improve speed and accuracy.

As a result, a program was created capable of processing video streams of different quality with great accuracy, classifying objects in the image and showing their locations. The advantage of the developed program is the ability to process exactly moving objects in the video stream, reduces the load on the processor and speeds up the system. The research of the developed system was also carried out and the work of the program was demonstrated on the example of video surveillance system.

Thesis project contains: 64 articles, 32 figures, 2 tables, 9 references to used sources.

Keywords: object identification, image processing, computer vision, OpenCV, CNN.

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
	А4	ІАЛЦ.045490.002 ТЗ	Програмні засоби	4		
			ідентифікації рухомих			
			об’єктів в відео-потоці			
			Технічне завдання			
	А4	ІАЛЦ.045490.003 ТП	Програмні засоби	2		
			ідентифікації рухомих			
			об’єктів в відео-потоці			
			Відомість технічного			
			проекту			
	А4	ІАЛЦ.045490.004 ПЗ	Програмні засоби	52		
			ідентифікації рухомих			
			об’єктів в відео-потоці			
			Пояснювальна записка			

Змін.	Арк.	№ докум.	Підпис	Дата	ІАЛЦ.045490.001 ОА		
Розробив		Бобров В.С.			Програмні засоби ідентифікації рухомих об'єктів в відео-потоці **Опис альбому**	Літ.	Аркуш 1 / Аркушів 2
Перевірив		Петрашенко А.В					
Консульт.						НТУУ «КПІ» ім. Ігоря Сікорського, ФПМ, КВ-63	
Н. контроль		Клятченко Я.М.					
Зав. каф.		Романкевич В.О.					

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
	А4	ІАЛЦ.045490.005 Д1	Архітектура розробленої	1		
			загорткової нейронної			
			мережі			
			Схема алгоритму			
	А4	ІАЛЦ.045490.006 Д2	Загальна схема проекту	1		
			Схема структурна			
	А4	ІАЛЦ.045490.007 Д3	Блок-схема розробленої	1		
			програми			
			Схема алгоритму			
	А4	ІАЛЦ.045490.008 Д4	Схема виявлення руху	1		
			Схема структурна			
		Диск CD-ROM	Текст пояснювальної			
			записки.			
			Графічний матеріал			

ЗМІСТ

1. НАЙМЕНУВАННЯ ТА ГАЛУЗЬ РОЗРОБКИ

Назва розробки: «Програмні засоби ідентифікації рухомих об'єктів в відео-потоці».

Галузь застосування: системи захисту, системи контролю.

2. ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки є завдання на дипломне проектування на здобуття першого (бакалаврського) рівня вищої освіти, затверджене кафедрою системного програмування і спеціалізованих комп’ютерних систем Національного технічного університету України «Київський Політехнічний Інститут імені Ігоря Сікорського».

3. МЕТА І ПРИЗНАЧЕННЯ РОБОТИ

Метою даного проекту є створення програмних засобів для ідентифікації рухомого об’єкту в відео-потоці.

4. ДЖЕРЕЛА РОЗРОБКИ

Джерелом інформації є технічна та науково-технічна література, технічна документація, публікації у періодичних виданнях та електронні статті у мережі Інтернет.

5. ТЕХНІЧНІ ВИМОГИ

5.1. Вимоги до програмного продукту, що розробляється

- кросплатформеність;
- гнучкість реалізаії;
- наявність виводу оброблених файлів;

- наявність вибору файлів для аналізу;

5.2. **Вимоги до апаратного забезпечення**

- Процесор: AMD A6;
- Оперативна пам’ять: 8 Гб;
- Наявність доступу до відео-камери;

5.3. **Вимоги до програмного та апаратного забезпечення користувача**

- Операційна система Linux, Windows;

6. ЕТАПИ РОЗРОБКИ

№ з/п	Назва етапів виконання дипломного проекту	Термін виконання етапів
1.	Вивчення літератури за тематикою проекту	10.03.2020
2.	Розроблення та узгодження технічного завдання	28.03.2020
3.	Аналіз існуючих рішень	11.04.2020
4.	Підготовка матеріалів розділів дипломного проекту	15.04.2020
5.	Підготовка звіту дипломного проекту	10.05.2020
6.	Передзахист дипломного проекту	20.05.2020

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
	А4	ІАЛЦ.045490.004 ПЗ	Програмні засоби	52		
			ідентифікації рухомих			
			об’єктів в відео-потоці			
			Пояснювальна записка			
	А4	ІАЛЦ.045490.005 Д1	Архітектура розробленої	1		
			загорткової нейронної			
			мережі			
			Схема алгоритму			
	А4	ІАЛЦ.045490.006 Д2	Загальна схема проекту	1		
			Схема структурна			
	А4	ІАЛЦ.045490.007 Д3	Блок-схема розробленої	1		
			програми			
			Схема алгоритму			

Змін.	Арк.	№ докум.	Підпис	Дата	ІАЛЦ.045490.003 ТП		
Розробив		Бобров В.С.			Програмні засоби ідентифікації рухомих об'єктів в відео-потоці **Відомість технічного проекту**	Літ.	Аркуш 1 / Аркушів 2
Перевірив		Петрашенко А.В.					
Консульт.						КПІ ім. Ігоря Сікорського, ФПМ КВ-63	
Н. контроль		Клятченко Я.М.					
Зав. каф.		Романкевич В.О.					

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
	А4	ІАЛЦ.045490.008 Д4	Схема виявлення руху	1		
			Схема структурна			
		Диск CD-ROM	Текст пояснювальної	1		
			записки.			
			Графічний матеріал			

Змін.	Арк.	№ докум.	Підпис	Дата	ДП.468300.003 ТП	Арк. 2

ЗМІСТ

OpenCV – бібліотека комп’ютерного зору і машинного навчання з відкритим кодом.

API – набір визначень підпрограм, протоколів взаємодії та засобів для створення програмного забезпечення.

CPU – функціональна частина комп'ютера, що призначена для інтерпретації команд.

CNN – загорткова нейронна мережа

Dataset – електронний набір даних.

ВСТУП

Не дивлячись на те, що проблема ідентифікації об’єктів існує давно, вона досі є актуальною. Початково, коли не було систем здатних аналізувати відео чи зображення, доводилося вручну переглядати фотокартки та відеозображення для виявлення рухомого чи просто об’єкту та подальшого визначення цього об’єкту. Але такий підхід відійшов на другий план, через стрімкий зріст кількості даних і їх накопичення. Тому постало питання, щодо систематизації і обробки великого об’єму даних.

Значним поштовхом у вирішенні цієї проблеми було створення глобальної мережі даних, яка дала можливість знаходити та обмінюватися ідеями і підходами, які вже були протестовані і не дали значних результатів в поставленому питанні. Одним з ефективних підходів було використання нейронної мережі, для автоматизованого знаходження і розпізнавання об’єкта після аналізу великої кількості подібних об’єктів.

На сьогодні, завдяки поширенню потужних комп’ютерів, доступність відеокамер здатних знімати відео високої якості та їх невеликої ціни та збільшенню попиту на технології які аналізують відеопотік розвиток алгоритмів знаходження і ідентифікації рухомих об’єктів стрімко йде в гору. Але проблеми все одно існують. В основному це пов’язано з нестачею даних чи інформації про сам об’єкт. Відомо, що об’єкт може бути лише частково на зображені, чи мати не типову форму, чи погане освітлення об’єкта, або об’єкт може бути перекритий перешкодами, які будуть заважати отримати повну характеристику об’єкта. Також, існує проблема значному обсягу обчислень для відеопотоку, через це невеликі і слабкі системи не можуть успішно класифікувати об’єкти. Для цього було розроблено алгоритми, які здатні

знаходити об’єкт за його переміщенням на відео-потоці. Це дає змогу аналізувати не кожний кадр, через що навантаження на систему стає меншим, і це призводить до більшої доступності таких систем.

Після того як всі проблеми будуть вирішені, з’являться системи які будуть здатні аналізувати навколишнє середовище і розпізнавати об’єкти з високою точністю, що дасть змогу покращити рівень життя в великих містах.

Автоматичне реагування на надзвичайні ситуації. Системи які можуть позбавити міста від заторів, завдяки здатності аналізувати трафік і коригувати світлофори. Виявлення потенційних злочинців, до того як вони зроблять злочин, що знизить рівень злочинності до мінімуму.

1.АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ

1.1 Метод Віоли-Джонса

Даний метод був розроблений і презентований ще в 2001 році Полом Віолой і Майклом Джонсоном, досі широко використовується в питаннях пошуку об’єктів в реальному часі.

Як відомо у кожного метода є своя основа,на якій будується подальша частина. В методі Віоли-Джонса за основу обрані ознаки Хаара(рис.1.1).

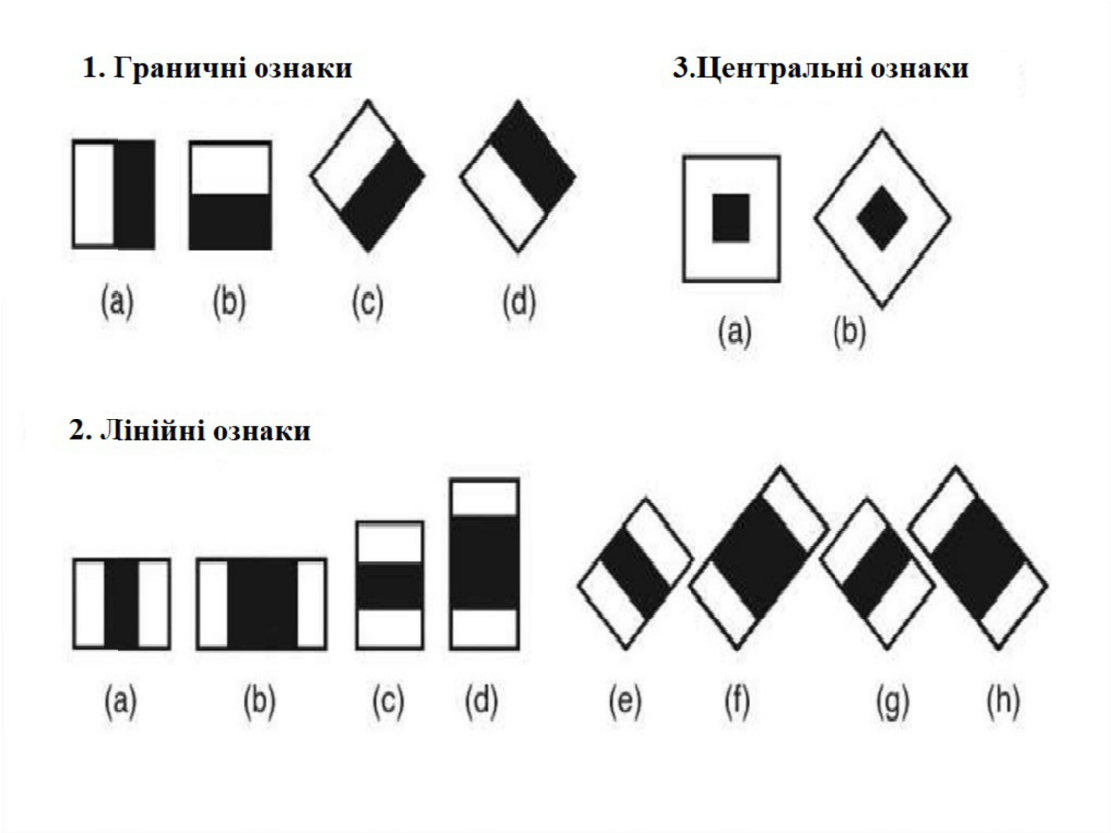

Рисунок 1.1 — Ознаки Хаара: 1. Граничні ознаки; 2. Лінійні ознаки; 3.Центральні ознаки

Спочатку використовувались ознаки без повороту,а для обчислення ознаки сума яскравості пікселів однієї області віднімалась з суми яскравості іншоїбласті. Але з розвитком метода було запропоновано ознаки з нахилом в 45 градусів. А також було запропоновано присвоювати кожній області вагу.

З пари пікселів на зображені важко дістати інформацію про область в якій вони знаходяться, але з двох ознак Хаара будується, наприклад, перший каскад по розпізнаванню обличчя(рис.1.2)

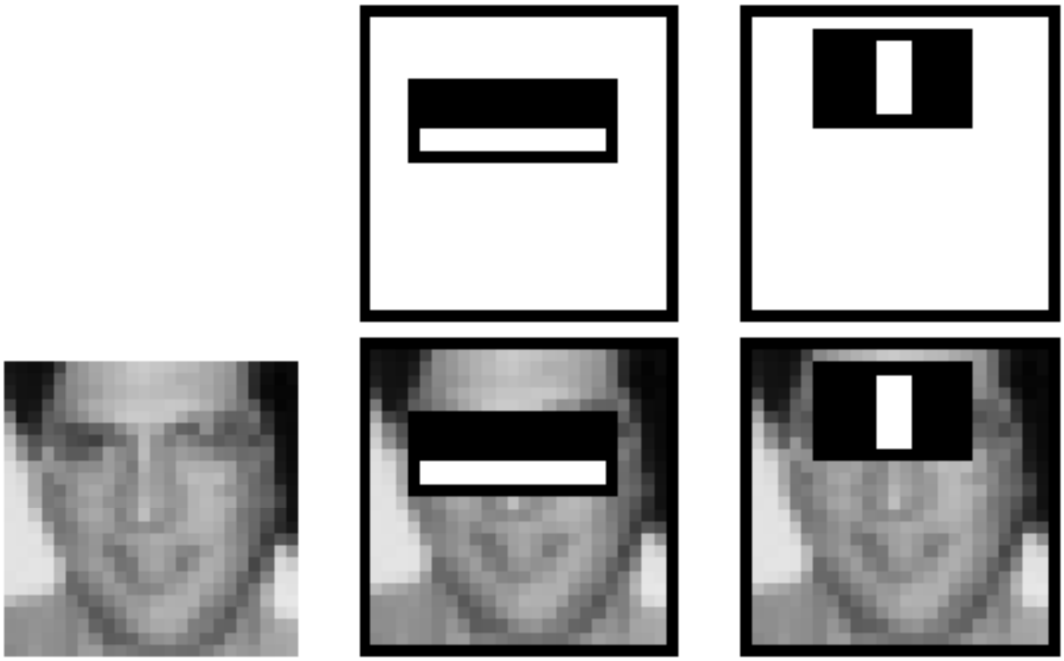

Рисунок 1.2 — Класифікація особливостей обличчя

Принцип роботи метода:

- По-перше створюється певний набір зображень (ознак) поділених за класами;

- далі беремо вікно сканування і вибрані ознаки;
- сканування поступово починає рух по зображенню ;
- при скануванні зображення в кожному вікні обчислюється приблизно 200 000 варіантів розташування ознак, за рахунок зміни положення і масштабу цих ознак;
- сканування відбувається послідовно для різних масштабів;
- всі знайдені ознаки потрапляють до класифікатору, який дає остаточну відповідь.

Даний алгоритм виявляє в кадрі, набір пікселів, які співпадають з підготовленими шаблонами які виглядають як білі та чорні прямокутників. Для кожного об’єкту, що аналізується потрібен свій набір шаблонів, які можна отримати після навчання системи на подібних об’єктах. Даний процес не швидкий і потребує правильної вибірки навчального набору даних.

1.2 Загорткова нейронна мережа

Згорткові нейронні мережі (CNNs) це нейронні мережі для вирішення таких питань, як розпізнавання об’єкта на зображенні та його класифікація.

Першим кроком CNN обробляє вхідні зображення після чого розбиває їх на певні категорії, наприклад, машина, людина, собака, кішка. Для комп’ютера зображення це масив пікселів. Цей масив виглядатиме так h x w x d , де h — це Height, w — це Width, d — це Dimension. Наприклад, зображення розмірність масиву пікселів якого 6x6x3 виглядатиме як на рисунку 1.3.

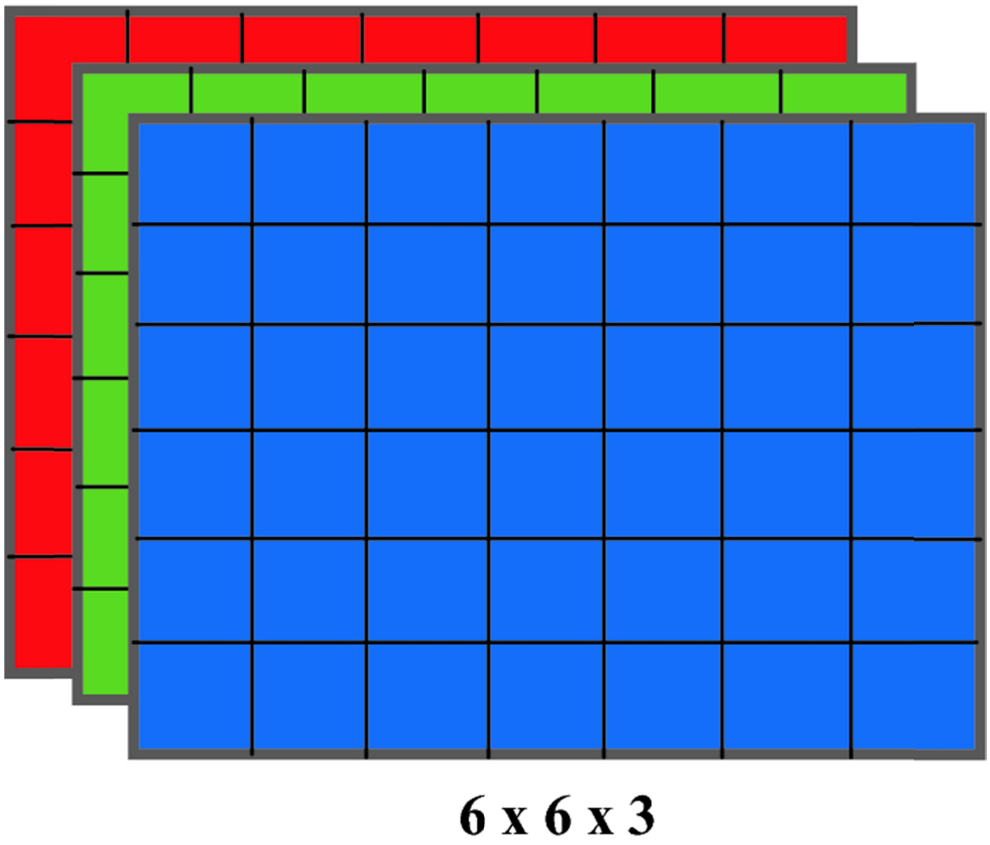

Рисунок 1.3 — Масив RGB матриці

На практиці зображення буде проходити через ряд загорткових шарів з різноманітними фільтрами (Kernals), субдискретизацію (pooling), і під кінці маємо повнозв’язний шар(FC), а в кінці буде застосована Softmax функція для класифікації об'єкта з ймовірними значеннями від нуля до одиниці(рис.1.4).

Softmax – це узагальнена логістична функція, що стискує K - вимірний вектор **z** із довільним значенням компенент до K -імірного вектора з дійсними значеннями компонент в області від 0 до 1 і сума координат дорівнює одиниці.

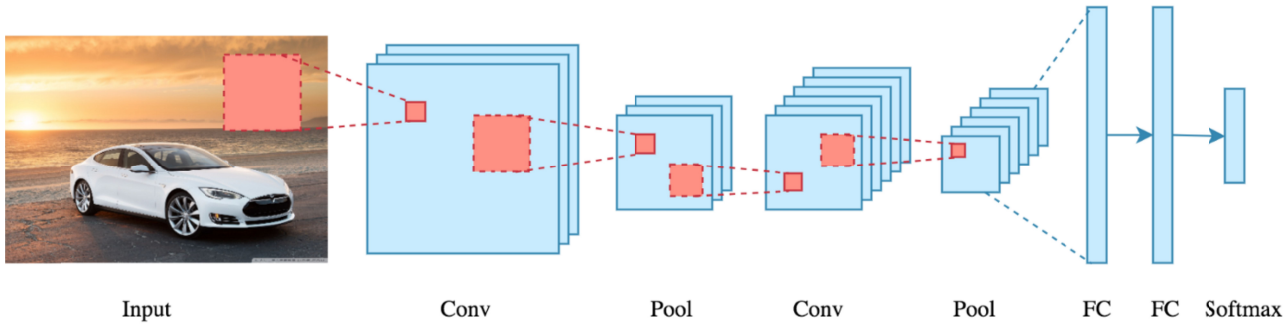

Рисунок 1.4 —Нейронна мережа з кількома загортковими шарами

Загортковий шар виконує операцію згортки і передає результат наступному шару. Шар цього типу виконує зменшення вхідної матриці зазвичай в 2 рази. Це математична функція яка приймає на вхід матрицю і фільтр. Наприклад, якщо на вхід було подано зображення 5 х 5 зі значеннями пікселів 1 або нуль і фільтр 3 х 3, то на виході мали б зображення 3 х 3(рис.1.5)

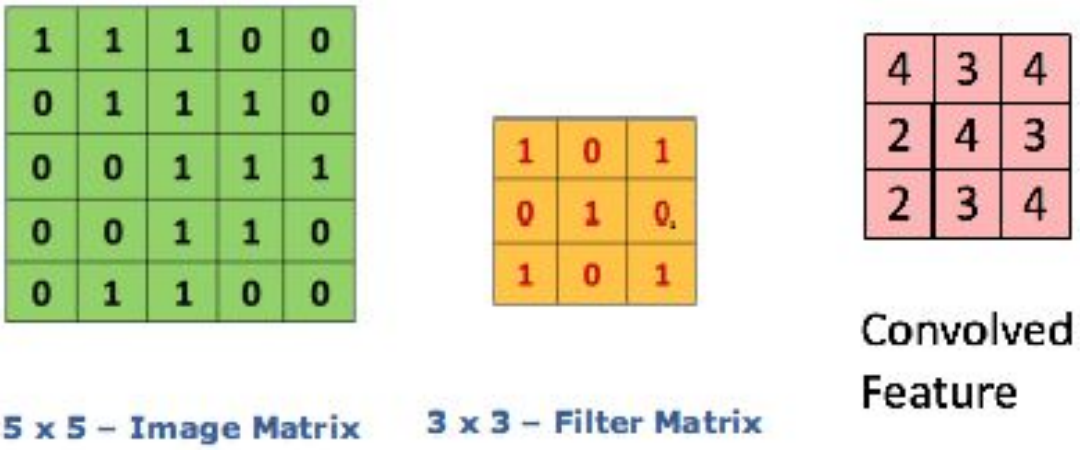

Рисунок 1.5 —Принцип роботи загорткового шару

Субдискретизований шар зменшить кількість параметрів, якщо зображення занадто великі. Зі зменшенням розміру важлива інформація не зникає. На рисунку 1.6 зображено приклад Max Pooling.

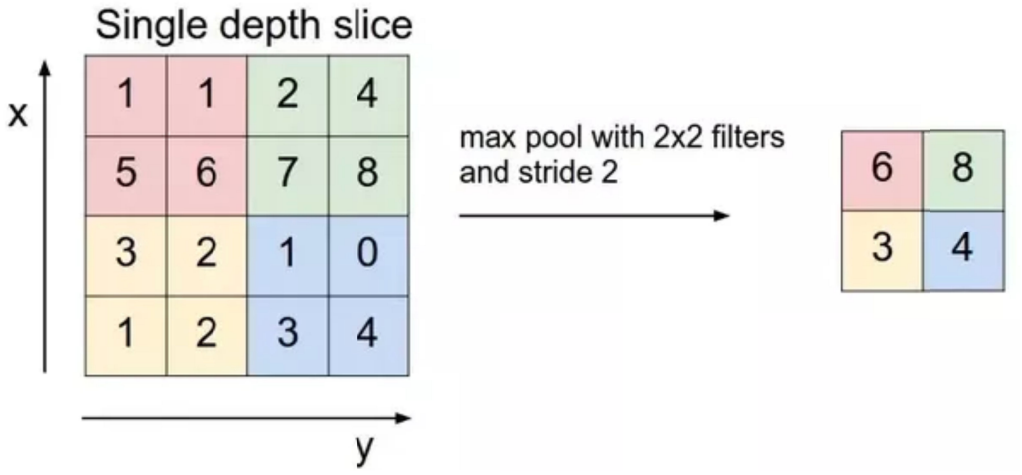

Рисунок 1.6 — Max Pooling

Повнозв’язний шар виглядає як нейронна мережа (рис. 1.7)

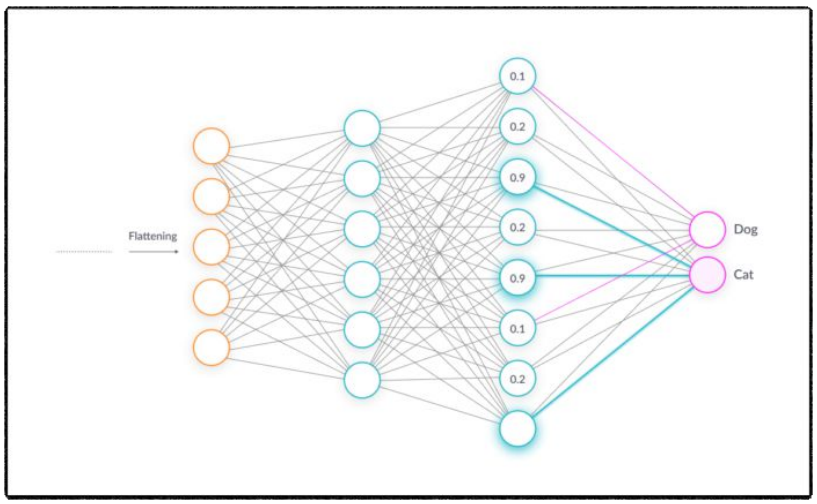

Рисунок 1.7 — Повнозв’язний шар

На цьому етапі , карти ознак матриці перетворюються на вектори і створюється повнозв’язний шар. І в кінці все об’єднується разом щоб створилась єдина модель і виконується функція активації щоб класифікувати вихідний шар(рис. 1.8).

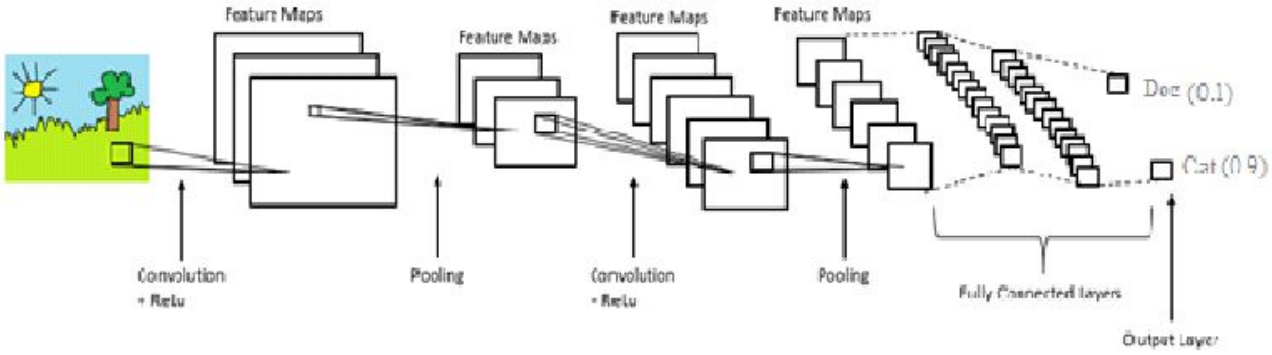

Рисунок 1.8 — Закінчена архітектура CNN

1.3 Виділення та аналіз контурів

Для того щоб знайти контури на зображенні необхідно зробити його монохромним. Для цього всі пікселі на зображені з ненульовим кольором будуть дорівнювати одиниці, а всі нульові зостануться нулями. Після цього робиться перевірка контурів на відповідність геометричним контурам об’єкта(рис1.9).

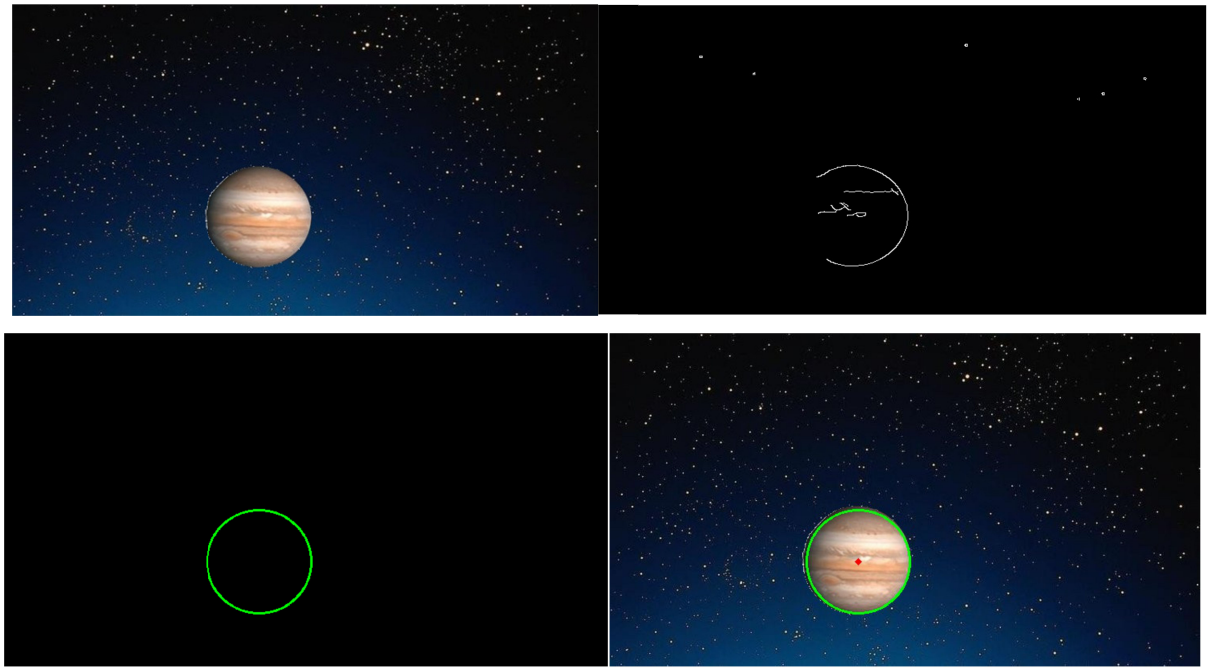

Рисунок 1.9 — Виділення та аналіз контурів

1.4 Шаблонний метод

Цей метод широко використовується для навігації роботів та контролю якості товарів. Головними недоліками цього метода є, оклюзія, знаходження зображень без чітких контурів, зміна освітлення і фону, зміна масштабу. Принцип роботи цього метода полягає в наступному — береться шаблонне зображення об’єкту,порівнюється з областями на основному зображенню(рис. 1.10), і в кінці області на основному зображенню які співпадають з шаблонним виділяються контуром.

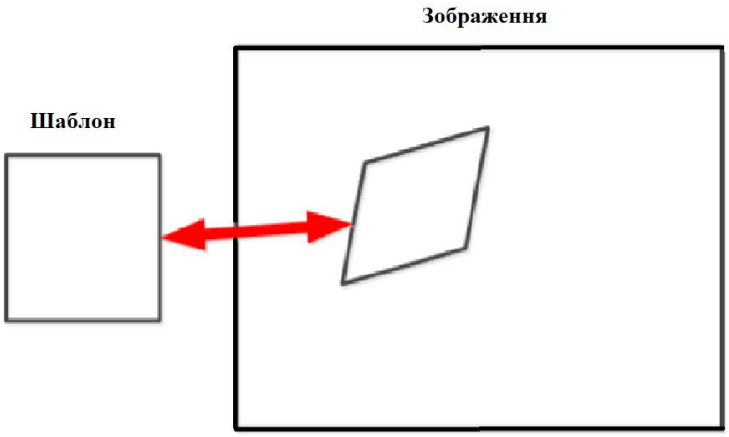

Рисунок 1.10 — Шаблонний метод

1.5 Пошук об’єкта за спеціальними точками

Спеціальними точками називають область на зображені, яка суттєво відрізняється від інших областей. Існує декілька методів знаходження цих точок, так звані блоби (blob, капля) — невеликі області однієї яскравості і чіткими границями, або кути. Для кожної спеціальної точки вираховують

дескриптор - спеціальну характеристику цієї точки. Потім ці дескриптори порівнюються. Цей метод використовується для створення панорам, слідкуванням за об’єктом, реконструкція трьохмірної моделі по його двомірних проекціях. Для того щоб знайти і виділити спеціальні точки використовують три алгоритми:

- SIFT
- SURF
- ORB

В алгоритмі SIFT для того щоб знайти спеціальні точки є побудовати піраміди гаусіанів (Gaussian) і знайти різницю гаусіанів (Difference of Gaussian, DoG).

Зображення яке було розмите гаусовим фільтром рахується за формулою:

$$L(x,y,\sigma) = G(x,y,\sigma) * I(x,y)$$

де L — значення гаусіана в точці з координатами(x,y), a sigma — радіус розмиття. G — гаусово ядро, I — значення початкового зображення ,* — операція згортки.

Різницею гаусіанів називають зображення, отримане шляхом по-піксельного віднімання одного гаусіана початкового зображення від гауссіана з іншим радіусом розмиття.

$$D(x,y,\sigma) = (G(x,y,k\sigma) - G(x,y,\sigma)) * I(x,y) = L(x,y,k\sigma) - L(x,y,\sigma).$$

Простором зображення, який масштабується, називається набір всіляких, згладжених деяким фільтром, версій вихідного зображення. Доведено, що

такий простір є лінійним, інваріантним щодо зрушень, обертань, масштабу, що не зміщує локальні екстремуми, і має властивість напівгруп.

Для нас важливо, щоб різна ступінь розмітки зображення гаусовим фільтром могла бути прийнята в тій чи іншій мірі за оригінальне отримане зображення.

Як правило, інваріантність масштабування досягається шляхом знаходження спеціальних точок для вихідного зображення, в різних масштабах. Для цього будують гаусову піраміду: масштабується простір ділиться на кілька відрізків - октави, а розмір області, що масштабується, яку займає наступна октава, в два рази більше, ніж розмір області, яку займає попередня октава. Крім того, при переході від однієї октави до іншої зображення зменшується, його розмір зменшується в два рази. Кожна октава покриває велику кількість гаусових зображень, тому тільки частина їх кількості N будується з певним кроком по радіусу розмітки. З цим же кроком будуються два додаткових гаусових зображень (N + 2), які виходять за межі октави.

Паралельно з будівництвом піраміди гаусіан будується піраміда різниць гаусіан. Відповідно,що кількість зображень, яке зберігається в цій піраміді, буде N + 1.

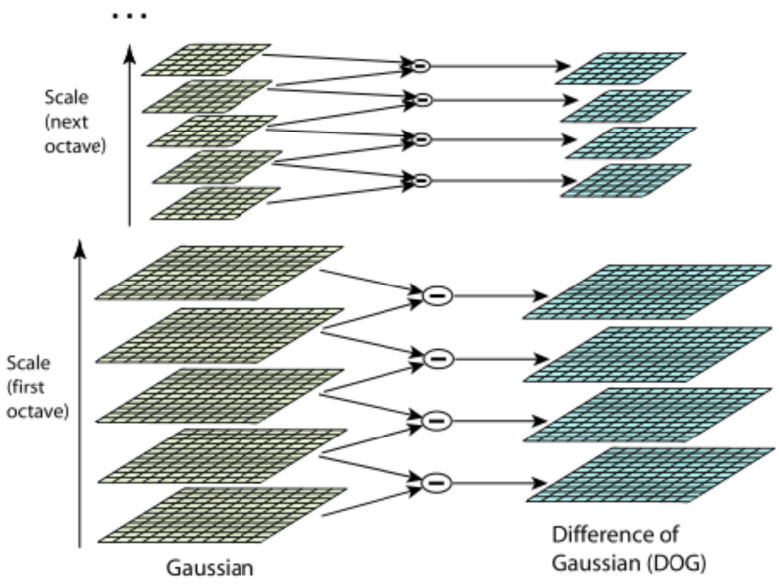

Рисунок 1.11 — зліва зображено піраміда гаусіанів, а праворуч - їх різниць.

На рис. 1.11 видно, що кожна різниця є виходом двох сусідніх гаусіанів, а кількість різниць на одиницю менше кількості гаусіанів, при переході до наступної октави, розмір зображення зменшується вдвічі.

Далі розглянемо локальний екстремум різниці гаусіанів. Для пошуку екстремумів скористаємося методом, схематично збереженим на рис. 1.12.

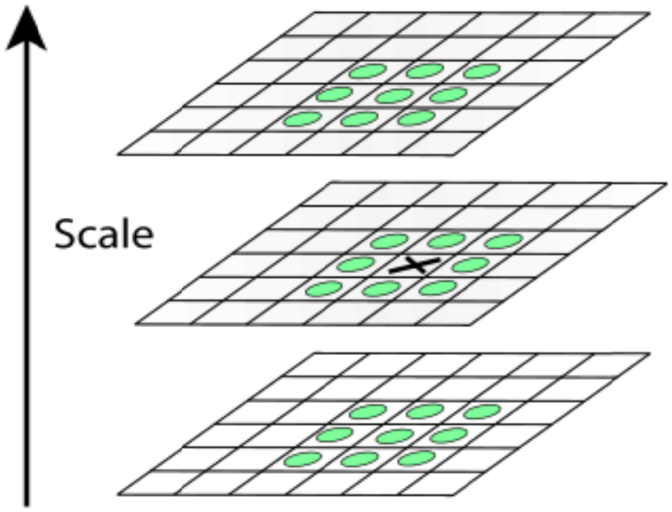

Рисунок 1.12 — Пошук екстремумів

Пошук локальних екстремумів проводиться на кожному ві з піраміди DoG. Кожна точка поточного DoG-зображення порівнюється зі своїми вісьмома сусідами і дев'ятьма сусідами в DoG, які в піраміді знаходяться на один рівень вище і нижче. Якщо ця точка більше (менше), ніж всі її сусіди, то вона приймається за точку локального екстремуму.

Тепер перейдемо безпосередньо до дескрипторів. В цілому, дескриптором може виступати будь-який об'єкт (аби він справлявся зі своїми функціями), але зазвичай дескриптором є якась інформація про околиці ключовий точки. Такий вибір зроблено в силу декількох причин: на маленькі області менший вплив роблять ефекти спотворень, деякі зміни (зміна положення об'єкту на зображенні, зміна сцени, перекриття одного об'єкта іншим, поворот) можуть не вплинути на дескриптор зовсім.

В такому методі як SIFT дескриптор виступає в ролі вектора. Як і напрямок ключовий точки, дескриптор обчислюється на гауссіані, найближчому за масштабом до ключовій точці, і виходячи з градієнтів в деякому вікні ключовий точки. Перед обчисленням дескриптора це вікно повертають на кут напрямку ключовий точки, чим і досягається інваріантність щодо повороту.

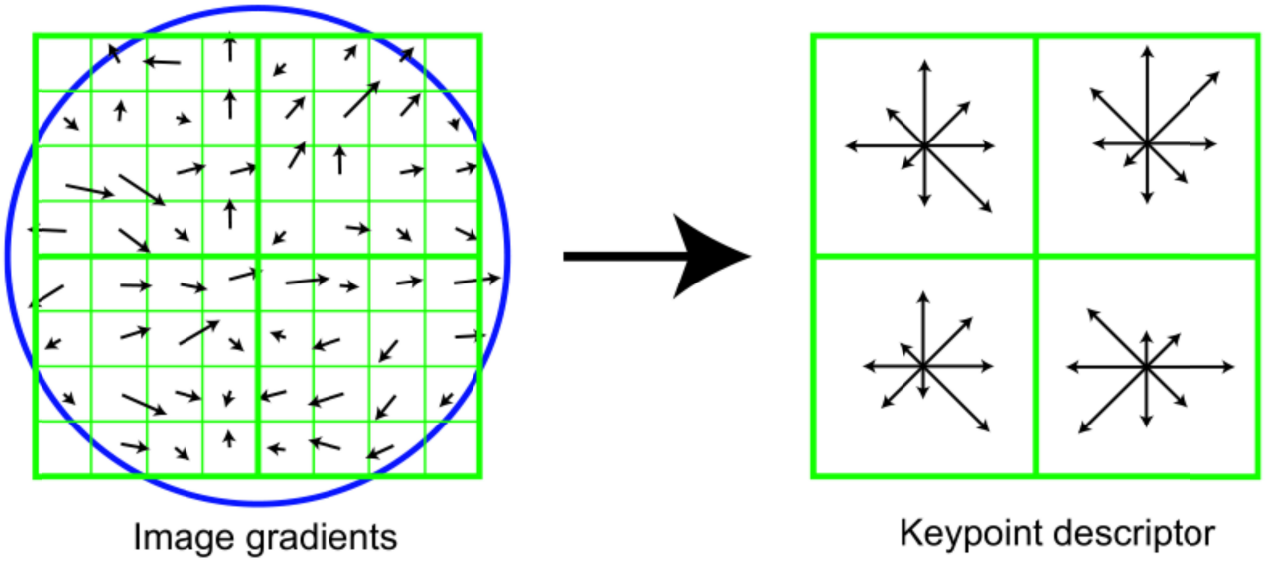

Рисунок 1.13 — Схема дескриптора

На рисунку 1.13 зображено градієнт вхідного зображення (ліва частина) і дескриптор (права частина), створений на основі цього зображення.. Коло на лівому зображенні, це вікно згортки гаусовим ядром. Sigma для цього ядра дорівнює половині ширини вікна дескриптора. До кожного градієнта у вікні дескриптора приписуються три ключові координати *x, y, n,* де *x* – це відстань по горизонталі до градієнта, *y* — відстань по вертикалі, *n*- відстань до градієнта в гістограмі.

Отриманий дескриптор нормалізується, після чого всі його компоненти, значення яких більше 0.2, урізаються до значення 0.2 і потім дескриптор нормалізується ще раз. У такому вигляді дескриптори готові до використання.

Метод Speeded Up Robust Features (SURF) добре вирішує задачі пошуку об'єктів. Цей метод широко використовують в біометричних системах аутентифікації людини па зображенню руки. На цьому прикладі розглянемо, як само працює цей алгоритм.

SURF алгоритм так само, як і SIFT знаходить спеціальні точки на вхідному зображенні та створює дескриптори для кожної точки . Дескриптори повинні бути інваріантними, щоб забезпечити коректну аутентифікацію, не дивлячись на те, в якому положенню знаходиться об'єкт.

Пошук спеціальних точок відбувається за допомогою матриці Гессе. Детермінант матриці Гессе (т.зв. гессіан) досягає екстремуму в точках максимальної зміни градієнта яскравості. Поворот ніяк не впливає на гессіан, але зміна масштабу впливає. Через це метод SURF використовує різномасштабні фільтри, щоб знайти гессіани. До кожної спеціальної точки розраховується масштаб і градієнт. За допомогою фільтрів Хаара обчислюється градієнт. Розмір фільтра дорівнює 4s (де s — масштаб спеціальної точки).

Після того, як всі спеціальні точки були знайдені , метод SURF присвоює кожній точці її дескриптор. Дескриптор виступає в ролі набору з 64 (або 128) чисел для кожної спеціальної точки. Через що, спеціальна точка - це максимум гессіан, виникає гарантія того, що коло цієї точки будуть ділянки з різними градієнтами, за рахунок цього, забезпечується відмінність дескрипторів для різних точок. Таким чином досягається інваріантність дескриптора щодо повороту. Розмір області, на якій вважається дескриптор, визначається масштабом матриці Гессе, що забезпечує інваріантність щодо масштабу.

ORB — це альтернатива SIFT і SURF в обчислювальної вартості, відповідної продуктивності і, в основному, патентам. Так як SIFT і SURF запатентовані, необхідно платити за їх використання, то за використання ORB цього робити непотрібно.

ORB — це злиття детектора ключових точок FAST і BRIEF дескриптора з великою кількістю різних модифікацій для підвищення продуктивності алгоритму. Спочатку використовується детектор спеціальних точок,а потім

застосовується вимір кута Harrisa для пошуку верхніх N точок серед них. Також створюється піраміда для отримання мульти-масштабованості. Але одна з проблем полягає в тому, що FAST не оріжнтування .

Також, алгоритм ORB обчислює інтенсивність центральної точки. Напрямок вектора від цієї точки кута до центральної точки дає орієнтування. Для покращення інваріантності обертання обчислюються моменти x і y, які повинні знаходитися в кругової області радіуса r, де r - розмір патча.

Тепер для дескрипторів ORB використовуйте BRIEF дескриптори. Але ми вже бачили, що BRIEF погано працює з ротацією. Тому ORB робить так, щоб "направляти" BRIEF відповідно до орієнтованих спеціальних точок.

Важливою властивістю BRIEF є те, що кожен біт має велику дисперсію і середнє значення, близьке до 0.5. Але як тільки він зорієнтований у напрямку спеціальних точок, він втрачає цю властивість і стає більш розподіленим.

Для того, щоб знайти об’єкт на зображені за допомогою спеціальних точок треба:

1. Знайти точки об’єкта і вирахувати їх дескриптори.
2. На зображенню так само шукаємо точки і вираховуємо дескриптори.
3. Перевіряємо чи співпадають дескриптори спеціальних точок об’єкта, з тими, що були знайдені на зображенню.
4. Якщо кількість співпадінь достатня,то цю область виділяємо.

Рисунок 1.13— Знаходження кутів на зображенні

Висновки до розділу

На сьогодні існує достатня кількість методів і алгоритмів для вирішення задач комп’ютерного зору. В цьому розділі було розглянуто методи які використовуються не тільки для знаходження об’єктів, а також для слідкування за ними, ідентифікації тощо. Надалі, в розробці програми, деякі з них будуть необхідні для вирішення помилок і покращення результатів.

2.ПРОБЛЕМАТИКА РОЗПІЗНАВАННЯ

2.1 Освітлення на зображенні

Освітлення може суттєво вплинути на обробку зображення і понизити якість і ефективність автоматизованої системи пошуку і розпізнавання об’єкту.

В основному при відсутності достатнього освітлення того чи іншого об’єкту, пошук і ідентифікація виконуються значно гірше, так сам і при занадто великої кількості світла. Це все пов’язано з тим, що при незначному світлі з’являються тіні, які порушують певні шаблонні риси об’єкта або зовсім приховують їх. Так само на зображення об’єкта впливає надмірне освітлення.

Щоб забезпечити ефективне знаходження і ідентифікацію об’єкта зображення повинно пройти через додаткові обробки, які зможуть відкоригувати освітлення до нормального стану(рис.2.1). Але щоб позбавитись від недоліків поганого освітлення завжди треба знаходити компроміс. Це пов’язано з тим, що при обробці картинки може з’явитись шум, зміниться глибинна різкість та витримка, але чим краще умови освітлення, тим легше знаходити баланс між цими параметрами.

Рисунок 2.1 — Обробка фотографії для зменшення освітлення.

2.2 Наявність перешкоджень

При пошуку об’єкта на зображенні найчастіше нічого не заважає знайти і виявити його, але коли перед об’єктом з’являється перешкода то цілісність порушується і це значно впливає на ефективність роботи алгоритмів.

Наприклад, якщо аналізувати обличчя, то такими перешкодами можуть стати окуляри, чи головний убір або обличчя частково перекрите рукою. Так само, якщо людина знаходиться за парканом, чи авто стоїть за дорожнім знаком (рис. 2.2) або за другім авто все це перешкоджає аналізу зображення і знаходження того чи іншого об’єкту.

Рисунок 2.2 — Дорожній знак перекриває авто.

2.3 Варіативність об’єктів

При ідентифікації рухомих об’єктів виникає складність в тому, що один і той самий об’єкт може виглядати по різному. Так само суттєво впливає кут огляду камери яка знімає. Для того щоб алгоритм зміг коректно навчитись розпізнавати об’єкт необхідно щоб в вхідних даних були всі можливі варіації цього об’єкту.

Тому ідентифікація рухомого об’єкту досить складне завдання і потребує достатньої кількості ресурсів для ефективного рішення.

Рисунок 2.3 — Варіативність зображення людини

2.4 Зміна об’єкту з часом або старіння об’єкту

Деякі об’єкти мають властивість змінюватись з часом, що негативно може впливати на процеси ідентифікації та їх знаходження. Якщо початковий ідентифікатор знаходиться в базі даних і не оновлювався з часом то різниця між двома зображеннями може бути суттєвою. Для цього необхідно оновлювати данні про ідентифікатор але існує інший підхід. Він полягає в тому, що існує можливість програмно застаріти об’єкт для виявлення варіантів того як він виглядає на даний момент.

Рисунок 2.4 — Процес старіння людини

2.5 Методи обробки зображень

2.5.1 Лічильник пікселів

Основна задача цього методу полягає в підрахуванні світлих або темних пікселів. Користувач має можливість за допомогою лічильника вибрати область

на екрані бажану область спостерігання,наприклад, там, де скоріш за все можна буде побачити людину. Камера одразу зреагує, і відправить інформацію стосовно кількості пікселів на обраному прямокутнику.

2.5.2 Бінарізація

Бінарізація — це процес перетворення вхідного зображення в бінарне. Це означає, що кожний піксель стає або білим або чорним. Умовно значення кожного пікселя кодуються як "0" або "1". Нульові значення частіше за все називають фоновими або фоном. Одиничні значення називаю переднім планом.

Найчастіше для збереження таких бінарних зображень використовують бітову мапу,в якій використовується один біт для одного пікселя.

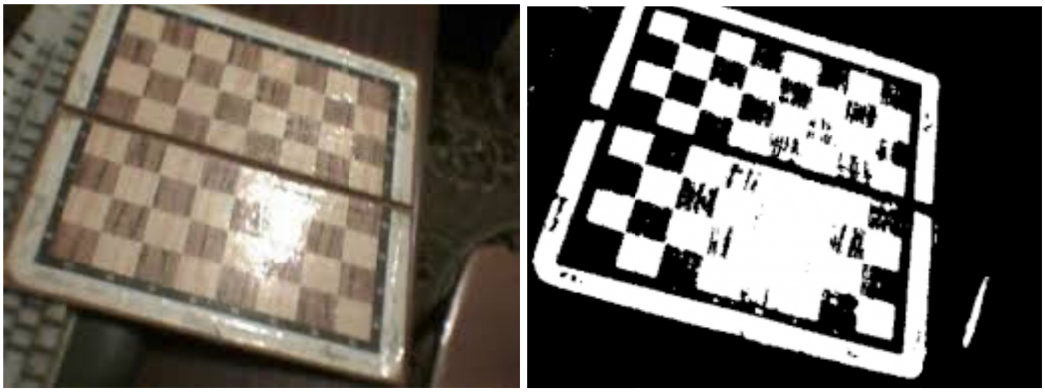

Рисунок 2.5 — Приклад бінарізації

2.5.3 Сегментация

Сегментація використовується для підрахунку і пошуку деталей. Головна мета сегментації — спрощення або зміна поданого зображення, щоб зробити аналіз легшим.

Зазвичай використовується для знаходження і виділення об’єктів та меж між нами на зображенні. Процес сегментації полягає в присвоєні кожному пікселю зображення таких міток, що пікселі які мають однакові мітки мають загальні візуальні характеристики.

Результатом сегментації зображення є сукупність сегментів,які разом охоплюють усе зображення, або набір контурів, вибраних із зображення.

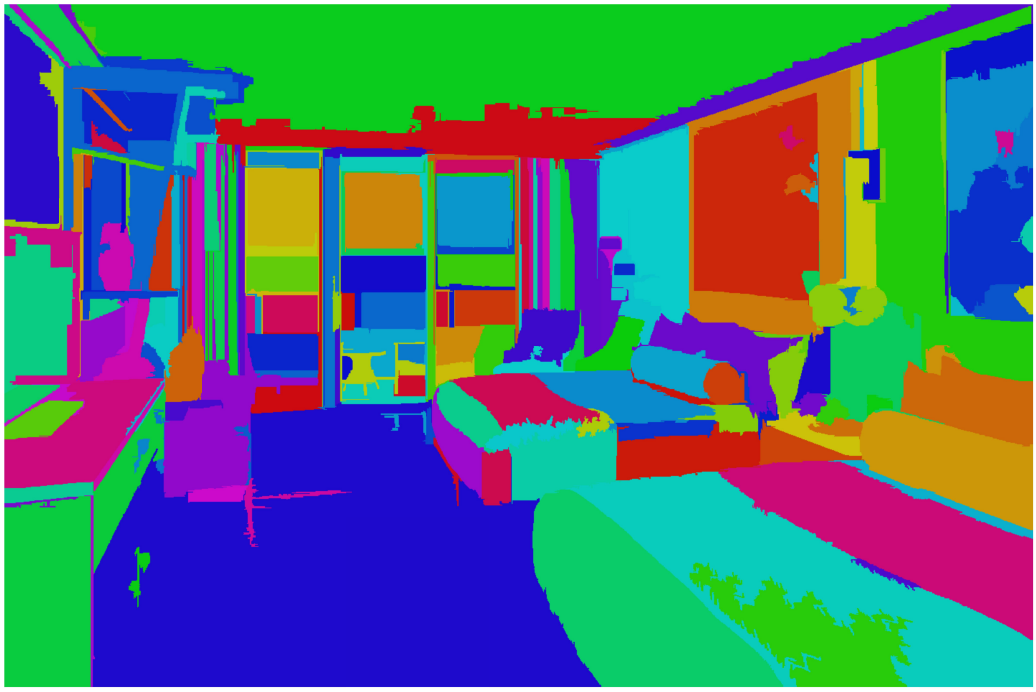

Рисунок 2.6 — Сегментація зображення кімнати

2.5.4 Читання штрих-кодів

Штрих-код — графічна інформація, що застосовується для надійності, маркування або упаковки продукції, що дає можливість зчитувати її іншим технічним приладам — послідовності чорно-білих смуг або інших геометричних фігур. У комп’ютерах штрих-коди використовуються для декодування 1D і 2D-кодів, розроблених для схем або машинного сканування.

2.5.5 Оптичне розпізнавання символів

Оптичне розпізнавання символів: автоматичне читання тексту, наприклад, серійні номери. Розпізнавання використовується для переведення книг та документів на електронну формат, також для автоматизації систем бухгалтерського обліку чи публікації тексту на веб-сторінці.

Оптичне розпізнавання тексту дозволяє редагувати текст, шукати слова чи фрази, зберігати їх у більш компактній формі, демонструвати чи друкувати матеріали, не втрачаючи якості, аналізувати інформацію та застосовувати електронний переклад, форматування чи перетворення мовлення до тексту.

Висновки до розділу

На сьогодні процес ідентифікації зображень не досяг ідеалу. Існує багато випадків коли програми розпізнавання працюють з зображенням на яких знаходиться неповна інформація, через що виникає помилка в аналізі і остаточній відповіді. Щоб покращити показники необхідно обробити вхідне зображення таким чином, щоб нестача інформації стала незначною. Для цього,на сьогодні існує багато алгоритмів і методів обробки зображень, більшість з яких буде використано і продемонстровано далі в проекті.

3.ПОРІВНЯННЯ ІНСТРУМЕНТІВ ДЛЯ РЕАЛІЗАЦІЇ ПРОГРАМИ

3.1 Бібліотека OpenCV

Для вирішення поставленого питання, необхідно було знайти бібліотеку в якій є модулі для обробки зображення та відео. Тому бібліотека OpenCV ідеально підійшла, тому що, вона відкрита і безкоштовна. Бібліотека написана на мові високого рівня C/C++ і містить в собі алгоритми для : слідування за об’єктами, сегментації об’єкта, розпізнавання жестів, знаходження подібностей і т.д.

Бібліотека включає в себе набір типових даних, функцій і класів для обробки зображень алгоритмами комп’ютерного зору. Основні модулі цієї бібліотеки це – cxcore, CV, Highgui, Cvaux, CvCam. Підтримка останнього модулю CvCam були припинена і в останніх версіях цей модуль вже відсутній.

Cxcore — це ядро, всі алгоритми і базові структури даних містяться саме в цьому модулі. З алгоритмів можна виділити:

- Запис чи відновлення структур даних в XML
- Базові функції 2D графіки
- Генератори випадкових чисел, математичні функції та лінійна алгебра
- Базові операції на багатовимірними массивами

Cvaux містить в собі функції експериментального характеру та функції які вже застаріли:

- просторовий зір, звукова калібрація, само калібрація
- пошук стерео-співпадінь
- знаходження і опис рис обличчя

Модуль Highgui служить для вводу і виводу зображень і відео (захоплення відео с камер і відео файлів, читання та запис статичних зображень), а також для створення користувацького інтерфейсу(функції для організації простого інтерфейсу).

Модуль CV містить алгоритми аналізу і обробки зображень:

- базові операції на зображені, такі як: фільтрація, геометричні перетворення, перетворення кольорових просторів і інші.
- анліз зображень (гістограми, пошук контурів, вибірка ознак, морфологія)
- аналіз руху і слідкування за об’єктом
- знаходження об’єктів, а також обличчя
- калібрування камер, елементи відновлення просторової структури.

Функціональність цієї бібліотеки доступна на різних мовах програмування: C, C++, Python, CUDA, Java. Також підтримуються основні операційні системи: MS,Windows, Mac, Android, iOS, Linux.

Описуючи ту чи іншу схему типової програми, яка вирішує питання комп’ютерного зору можна виділити деякий шаблон.

На початку роботи програми відбувається захоплення відеопотоку або зображення (модуль Highgui). Це відбувається в реальному часі або з записаного відео .

Наступний крок — позбавлення шуму , нормалізація освітлення шляхом видалення зайвих бліків і тіней, вирівнювання контрасту. Це все обробка зображення для подальшої роботи з ним. З цією задачею справляється модуль imgproc. Після обробки зображення за роботу беруться модулі features2d і imgproc. Для просто прикладу візьмемо такий приклад, коли стає питання щодо захоплення об’єкта і слідкування за ним, необхідно виконати пошук спеціальних точок, за якими буде легко вести нагляд.

Далі підключаються модулі video, calib3d, ml, videostab. Вони виконують основні задачі по знаходженню об’єкта, аналізу його структури, коригування камери і інші.

В кінці модуль ml розпізнає і приймає рішення стосовно отриманого зображення. Чи був знайдений той чи інший об’єкт, який це об’єкт, або які саме цифри і букви знаходяться на номерному знаку авто.

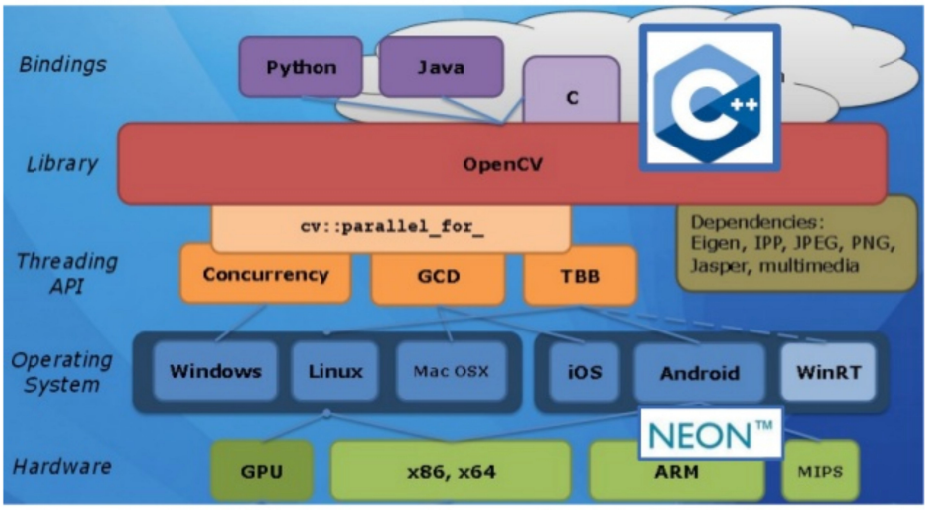

Рисунок 3.1 — Функціональність і взаємозв’язок бібліотеки OpenCV

3.2 Попередня обробка вхідного зображення

Функціонал бібліотеки для попередньої обробки зображень дуже великий. Розглянемо методи і функції, які використовують в області ідентифікації і знаходження рухомих об’єктів.

Перший метод, який буде розглянутий, знаходження і виділення контурів.

По-перше необхідно перетворити зображення на монохромне. Для цього необхідно скористатися функцією void cvtColor(const Mat& src, Mat& dst, int code) де src — вхідне зображення для обробки, dst — вже оброблене зображення, code — код операції конвертації.

Далі необхідно додати розмиття для того, щоб позбавитись не потрібних ребер. Для цього використовуємо функцію void GaussianBlur(InputArray src, OutputArray dst, Size ksize, double sigmaX, double sigmaY=0, int borderType=BORDER_DEFAULT) де src — вхідне зображення, dst — вихідне зображення, ksize— розмір Гаусового ядра, sigmaX— стандартне відхилення ядра в X напрямі, sigmaY— стандартне відхилення ядра в Y напрямі, borderType — піксельний метод екстраполяції.

І на кінець виконуємо функцію cvCanny(const CvArr* image, CvArr* edges, double threshold1,double threshold1, int aperture_size) де image — зображення для обробки, edges — зображення для збереження знайдених границь, threshold1 — поріг мінімуму, threshold2 — поріг максимуму, aperture_size — розмір для оператора Соболя. Функція виконує пошук границь за алгоритмом Кані.

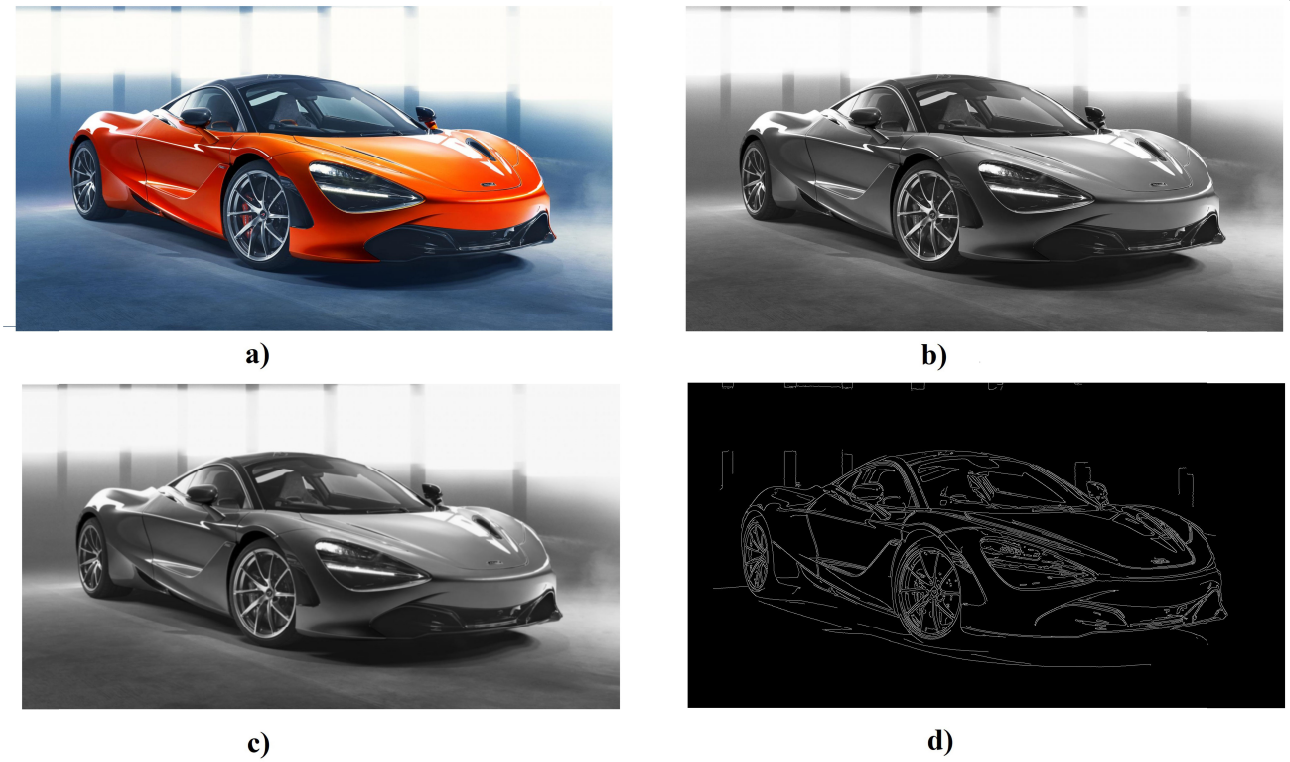

Рисунок 3.2 — Поетапний процес знаходження контурів: a) Оригінальне зображення; b) Монохромне; c) Розмите монохромне зображення; d) Кінцевий результат.

3.3 Пошук об’єктів на зображенні

Для прикладу буде застосований метод знаходження спеціальних точок.

Для початку проаналізувати об’єкт який будемо шукати на спеціальні точки, які далі ми і будемо шукати на вхідному зображенню. Все необхідне для реалізації цього метода знаходиться в модулі feature2d. Завдяки йому ми зможемо знайти точки, створити їх дескриптори, та застосувати алгоритми їх порівняння. Для

виявлення спеціальних точок можна застосувати один з трьох алгоритмів:

- SIFT

- SURF
- ORB

У кожного алгоритму є свої плюси та мінуси,дивлячись від типу зображень. SIFT і SURF запатентовані, тоді як ORB є безкоштовним. За допомогою функції ORB_create() створюємо детектор. Після чого знаходимо точки на обох зображеннях за допомогою функції orb.detect(). Після чого обчислюємо дескриптора для кожної точки за допомогою orb.compute().

Далі необхідно знайти схожі дескриптори на кожному з зображень. Це можна зробити за допомогою BFMatcher(). І останнім кроком необхідно намалювати ці точки і їх зв’язок.

Рисунок 3.3 — Зв’язок між двома зображеннями

3.4 Бібліотека TensorFlow

Tensorflow часто використовується для вирішення проблем глибокого навчання, а також для навчання і оцінки процесів аж до впровадження моделі. Крім машинного навчання, TensorFlow можна також використовувати для моделювання будівель на основі приватних похідних рівнянь. Тому він вважається універсальним інструментом для навчання інженерів верстатобудування.

Для підвищення продуктивності бекенда TensorFlow був написаний на мові C, який працює неймовірно швидко. Однак API доступний як на C ++, так і на Python, найбільш широко використовувану мову для машинного навчання, а також на інших мовах, таких як Java, Go, Haskell, JavaScript, Swift та багатьох інших. TensorFlow також був адаптований для роботи на різних платформах; крім Linux, MacOS і Windows, ви можете використовувати моделі Tensorflow на iOS, Android і Raspberry Pi.

Найбільшою перевагою TensorFlow є те, що він дозволяє використовувати API різного рівня для досягнення необхідного ступеня абстракції. Наприклад, створення нових моделей за допомогою API високого рівня Estimators, який корисний для навчання, прогнозування і функцій розгортання. Однак API низького рівня дає більше свободи для експериментів з архітектурою моделі або гіперпараметрами.

TensorFlow виконує математичні операції над великими багатовимірними масивами чисел, які можна узагальнити як тензори. Щоб краще зрозуміти, як працює TensorFlow, уявіть собі спрямований граф з тензорами по його краях і вузлами, представленими у вигляді математичних операцій між тензорами. Обчислення на цьому графі виконуються в рамках сеансів. Така абстракція походить від концепції низькорівневого програмування, так звана "лінива

оцінка", при якій дані і операції ініціюються, а потім обчислюються через сеанс. Такий підхід дозволяє використовувати CPU разом з GPU, а також кілька GPU для паралельних обчислень, що робить все процеси, особливо навчання, значно швидше.

3.5 YOLOv3

YOLO або Look You Only Once — архітектура CNN, яка використовується для розпізнавання великого різноманіття об’єктів на зображеннях.

Головна особливість цієї архітектури порівняно з іншими полягає в тому, що більшість систем застосовують CNN кілька разів до різних регіонів зображення, в YOLO CNN застосовується один раз до всього зображення відразу. Мережа ділить зображення на своєрідну сітку і пророкує bounding boxes і ймовірності того, що там є шуканий об'єкт для кожної ділянки. Перевагами даного підходу є те, шо мережа сканує все зображення відразу, і враховує контекст при знаходженню й розпізнаванню об'єкта. Так само YOLO в 1000 разів швидше, ніж R-CNN і близько 100х швидше ніж Fast R-CNN.

YOLOv3 — це вдосконалена версія архітектури YOLO. Вона складається з 106-ти згорткових шарів. Основна особливість YOLOv3 полягає в тому, що на виході є три шари кожен з яких розрахований на виявлення об'єктів різного розміру.

Також існує архітектура YOLOv3-tiny — обрізана версія архітектури YOLOv3, яка складається з меншої кількості шарів (вихідних шарів всього 2). Вона більш гірше пророкує дрібні об'єкти і призначена для невеликих датасетів. Але, через спрощену будову, ваги мережі займають невеликий обсяг пам'яті (приблизно 35

Мб) і вона видає більш високий FPS. Тому така архітектура добре підходить для використання на мобільних пристроях.

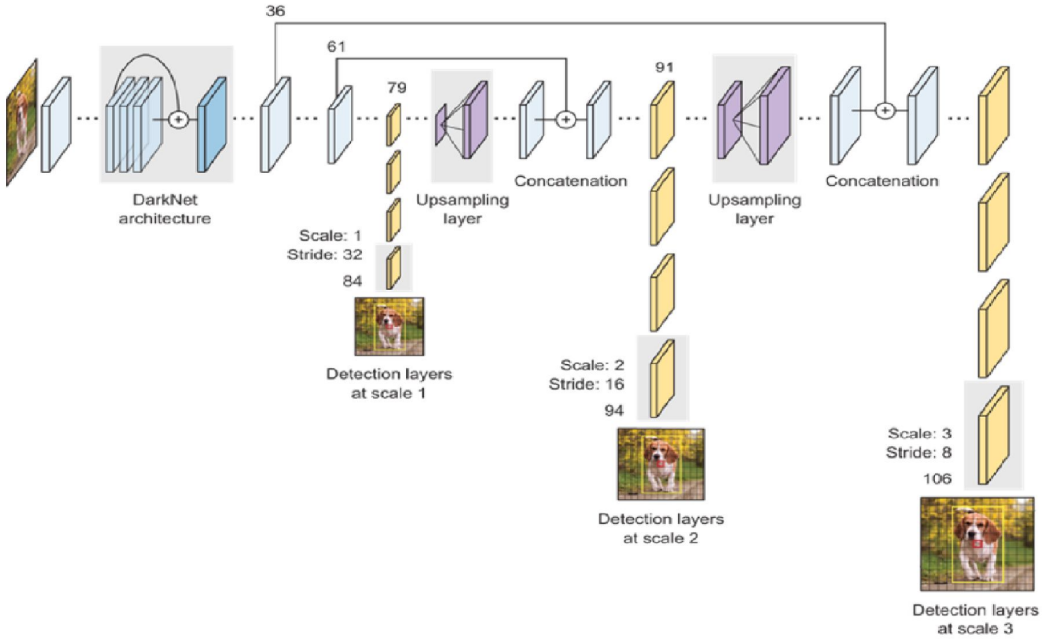

Рисунок 3.4 — Архітектура нейронної мережі в YOLOv3

3.6 Порівняння готових моделей CNN з для визначення об’єктів

Для того щоб порівняти моделі, я написав невелику програму на вхід якої завантажується зображення і моделі нейронних мереж. В якості критеріїв оцінювання я вибрав: правильність знайденого об’єкту і точність оцінки знайденого об’єкту. Для тестування було проаналізовано 50 зображень людини, собаки і автомобілів.

В якості моделей я обрав AlexNet (Caffe framework) , GoogleNet(Caffe framework) , ResNet-50 (Caffe framework), YOLOv3(Darknet framework), MobileNetSSD(Caffe framework)

Назва моделі	Розмір зображення	Зобр. людини	Зобр. собаки	Зобр. машин	Середня точніст
ResNet	224x224	94,8%	87,5%	91,2%	91.16%
YOLOv3	416x416	98,9%	96,7%	97,6%	97.3%
MobileNetSSD	300x300	91,1%	85,3%	89,9%	88.76%

Таблиця 4.1 – Порівняння точності існуючих моделей

Найбільший показник точності у YOLOv3 тому для розробки власної поделі я буду використовувати цю модель.

Висновок до розділу

Для розробки програми необхідно обрати правильну платформу і бібліотеку, здатну вирішувати задачі комп'ютерного зору. Найпопулярніші на сьогодні бібліотеки OpenCV і Tensotflow добре підходять для цього. Також в цьому розділі було показано бібліотечні функції бібліотек для обробки зображень. Окрім цього, було проведено дослідження щоб обрати найточнішу модель CNN для подальшої роботи з нею.

4 АРХІТЕКТУРА РОЗРОБЛЕНОЇ СИСТЕМИ

Після проведених досліджень було прийнято рішення обрати мову програмування Python, а також розбити систему на декілька модулів, для того, щоб була можливість змінювати конфігурації модулів без зміни основної логіки програми.

Модуль детектор відповідає за:

- Виявлення об’єкту на зображенні;
- Знаходженню його координат на кадрі;
- Оцінки ймовірності того, наскільки відсотків об’єкт відповідає заданому класу.

Найкращим варіантом, для вирішення цих задач є загорткові нейронні мережі. Для своєї програми я обрав YOLOv3, яка підтримує transfer learning. Transfer learning надає можливість використовувати накопичений досвід в вирішенні одного питання для вирішення подібного. Підхід цієї системи дає можливість оброблювати зображення значно швидше, через те, що вона виявляє всі місця на зображенні, де може бути об’єкт за одне проходження.

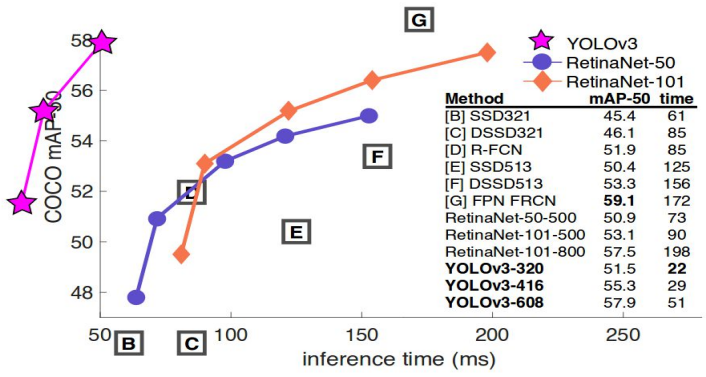

Рисунок 4.1 — Швидкодія і точність YOLOv3 у порівнянні з конкурентами

Також великою перевагою є те, що код цієї системи відкритий і що вже є натреновані системи для розпізнавання різних наборів даних.

Так як більшість натренованих систем виконують класифікацію більше ніж 100 класів було вирішено натренувати системи власноруч.

По-перше треба було завантажити dataset з зображень на яких система буде тренуватись. Для цього я скористався відкритою базою зображень Google в якій вже зібрані більше 9 мільйонів зображень для тренування. Відмінність таких зображень в тому, що вони підібрані по класам, і кожний клас на зображенні виділений прямокутником. До кожного зображення прикріплений текстовий файл, в якому міститься інформація про те, який саме це клас і координати прямокутника, в якому міститься об’єкт. Потім всі файли ми збираємо в єдину директорію, шлях якої подається на вхід нашої мережі.

Процес тренування був достатньо довгий, через те що було завантажено більше 1Гб зображень.

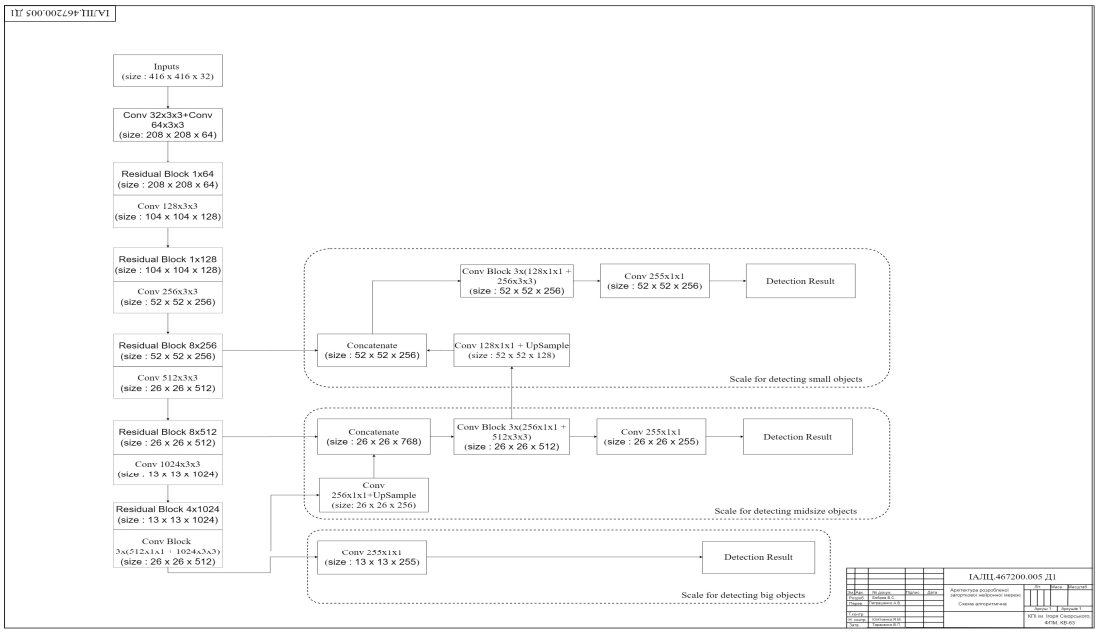

Рисунок 4.2 — Схема розробленої CNN

Після тренування з’являються файли ваг і конфігурації які надалі я буду використовувати для знаходження об’єктів.

Для того щоб коректно оброблювався кожен кадр , я попередньо зменшую його розмір, щоб модель нейронної мережі правильно працювала.

Далі для того щоб розуміти скільки кадрів буде опрацьовано, я додав функцію для вираховування кількості кадрів вхідного відео-потоку.

Наступним кроком необхідно було реалізувати цикл, щоб проаналізувати кожний кадр. Для цього я використав нескінченний цикл (while True:) , а також два виходи з нього. Перший варіант, при якому програма закінчує свою роботу – це коли користувач нажимає space на клавіатурі . Для другого варіанту було додано перевірку на захват кадру, якщо нічого не захопилось то відео скінчилось, а отже треба припинити роботу програми.

Після того , як нейронна мережа виявила всі об’єкти на зображенні треба було відсортувати їх. Для цього я додав вхідну змінну в діапазоні від 0 до 1, яка відповідає за контроль впевненості , щоб порівнювати її з значенням знайдених об’єктів. Якщо впевненість в тому що знайдений об’єкт відповідає своєму класу більша ніж задане значення то кадр оброблюється далі. Коли перевірка пройдена, створюється список в який додаються області всіх знайдених об’єктів.

Для того щоб позбавитись великої кількості малих прямокутних областей на одному і тому ж об’єкті, необхідно виділити велику прямокутну область в яку попадають всі малі. Для цього необхідно було відфільтрувати зображення. Тому я вибрав алгоритм Кенні щоб подавити всі малі прямокутники. В OpenCV є функція NMSBoxes() за допомогою якої я і фільтрував зображенння.

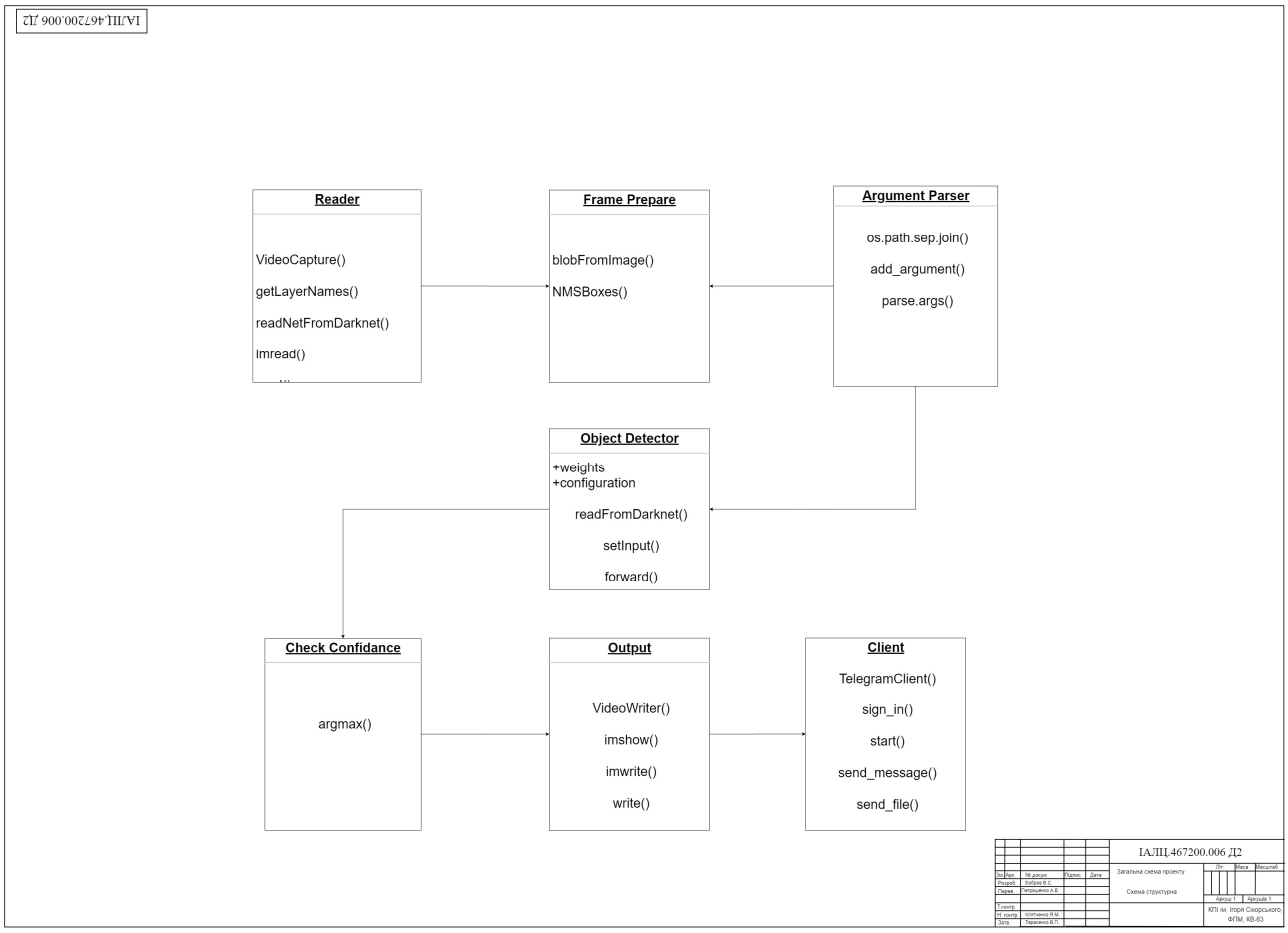

Рисунок 4.3 – Структурна схема розробленої програми

4.1 Тестування розробленої системи

Для того щоб протестувати розроблену систему, необхідно було додати функцію для запису вихідного зображення з інформацією знайдені об’єкти. Для цього я скористався функцією VideoWriter() з бібліотеки OpenCV. В якості результатів роботи програми я зробив скріншоти з вихідного відео-потоку з різницею часу в 5 секунд (див рис. 4.4)

Рисунок 4.4 – Результати розробленої системи

Під час роботи програми в середньому завантаженість на CPU 80% , а об’єм пам’яті який використовує програма досягав приблизно 600Mb. Такі показники мене не влаштовували, через те що програма занадто важка, і час на обробку відео-потоку, в якому 1173 кадрів, займав 4788 секунд (79.8 хвилини). Тому необхідно було модифікувати розроблену програму.

Для покращення роботи програми, було вирішено створити модуль який би фіксував рух на зображенні. Перед тим, як зображення перейде до детектора робиться перевірка на відмінність двох кадрів.

Спочатку за допомогою бібліотечних функцій OpenCV фільтрується зображення. За допомогою функції cvtColor() програма робить з поточного кадру, зображення з сірим тоном, після чого розмиває його за допомогою функції GaussianBlur().

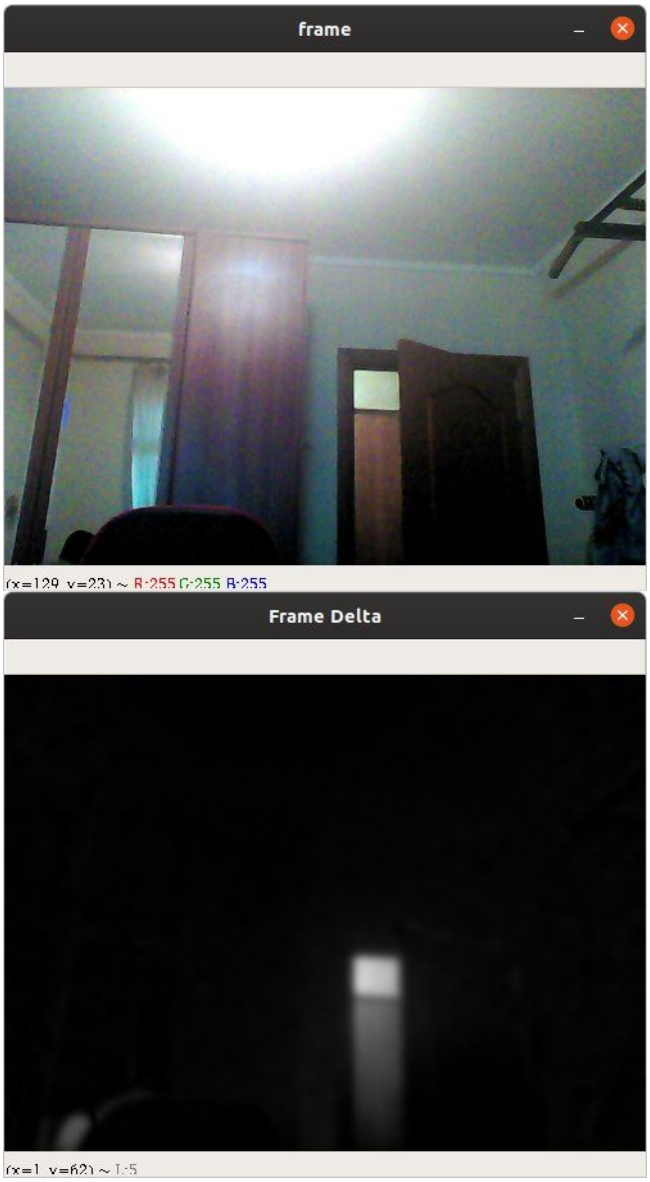

Рисунок 4.5 – фільтрація поточного кадру

Це відфільтроване зображення буде використовуватись для порівняння з наступним. Наступний кадр проходить такий самий процес, після чого різниця між цими кадрами рахується і записується за допомогою функції absdiff().

Здавалось, що цього буде достатньо для фіксації руху, але після тесту я зрозумів, що програма реагує навіть на невелику різницю в кадрі, через це необхідно було модифікувати програму далі. Для вирішення цієї проблеми до різниці двох кадрів застосовується процес бінарізації зображення. Після чого виділяються контури області руху. Останнім кроком перевіряється розмір цієї області, якщо вона занадто мала, то програма не фіксує руху.

Тепер програма не така важка. Через те, що детектор починає свою роботу тільки після появи рухомого об’єкту, середня завантаженість на CPU зменшилась, а швидкість роботи програми збільшилась.

	Початкова ситстема	Модифікована
Кілкість кадрів які надійшли до детектора	1173	919
Середнє навантаження на CPU %	80%	64%
Час (сек)	4563	3574
Середній об’єм зайнятої пам’яті(Mb)	583	512

Таблиця 4.2 – Результати порівняння двох систем

Як можна побачити з таблиці 4.2 модифікований алгоритм показує кращі результати за початковий. А найголовніше, що тепер програма реагує тільки на рухомі об’єкти.

4.2 Практичне використання створеної програми

Як приклад було вирішено обрати концепцію розумного будинку в якому є детектори руху і камери відео-спостереження. Якщо злочинець потрапляє до дому, то система одразу відправляє власнику повідомлення про те, що в дім потрапили посторонні.

В якості онлайн клієнта було обрано Telethon. Telethon — це клієнтська бібліотека для API Telegram. Найбільш відомим з API Telegram є його API-інтерфейс Bot, API-інтерфейс на основі HTTP для розробників, взаємодіючих з бот-платформою. Bot API дозволяє розробникам керувати ботами Telegram, наприклад, отримувати повідомлення і відповідати іншим користувачам. Крім Bot API, є і сам Telegram API. Це API, який використовується додатками Telegram для всіх ваших дій в Telegram. Кілька таких дій: перегляд ваших чатів, відправка і отримання повідомлень, зміна зображення на дисплеї або створення нових груп. За допомогою Telegram API можна робити все, що можете, в додатку Telegram програмно. Також Telegram API набагато складніша, ніж Bot API. Доступ до API-інтерфейсу бота отримується через HTTP-запити зі стандартними корисними навантаженнями JSON, форми або рядка запиту, в той час як Telegram API використовує власний формат корисного навантаження і протокол шифрування (рис. 4.6). Тому було обрано саме Telegram API замість

Bot Api.

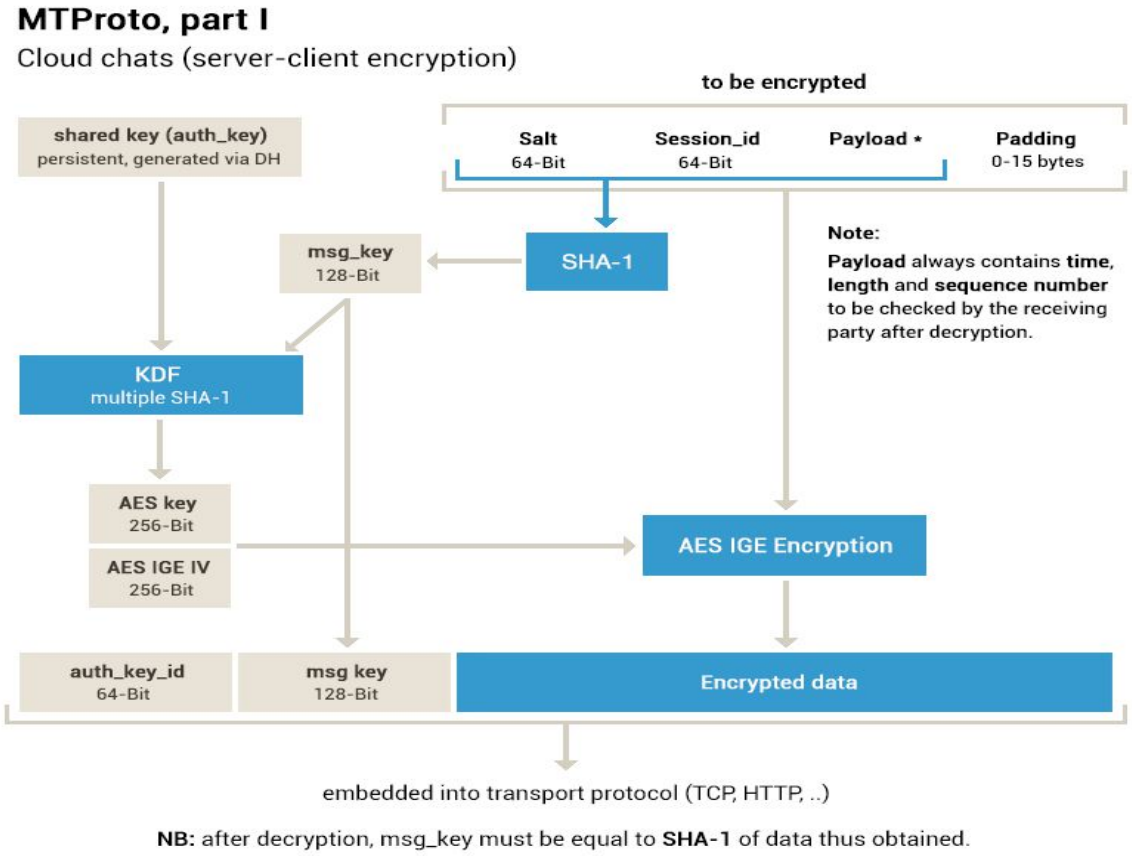

Рисунок 4.6 — Схема протоколу шифрування Telegram API

При запуску програми на вхід, замість відео-потоку з файлу подається відео-потік з зовнішнього пристрою(камеру краще за все поставити перед вхідними дверима.). Далі аналізується вхідний кадр так, як було описано в розділі 4.1.

За допомогою функцій imwrite() захоплений кадр з рухомим об’єктом зберігається на носії і за допомогою онлайн клієнта фотографія відправляється в обраний власником діалог (рис 4.7)

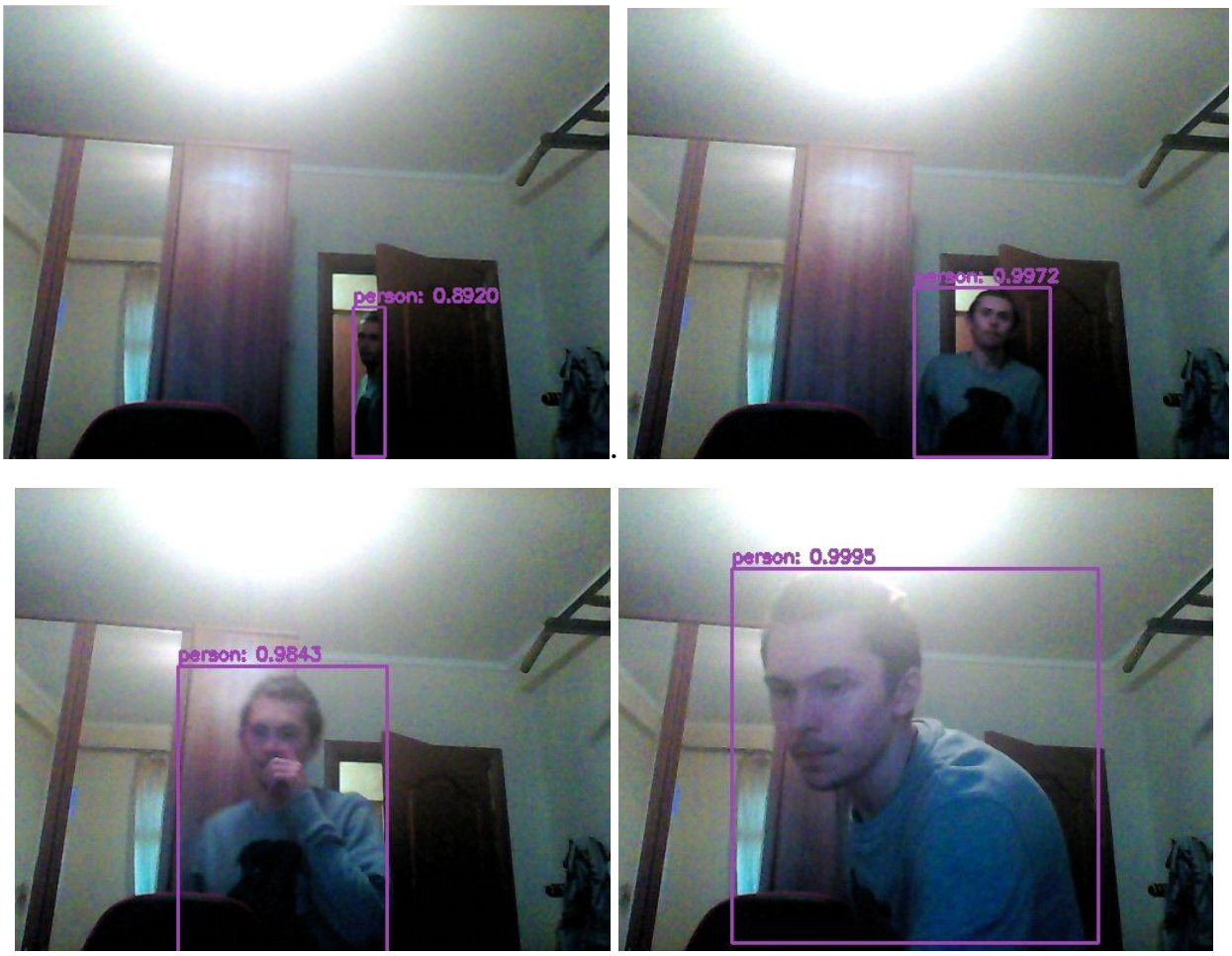

Рисунок 4.7 – Захоплені кадри з людиною яка увійшла в кімнату

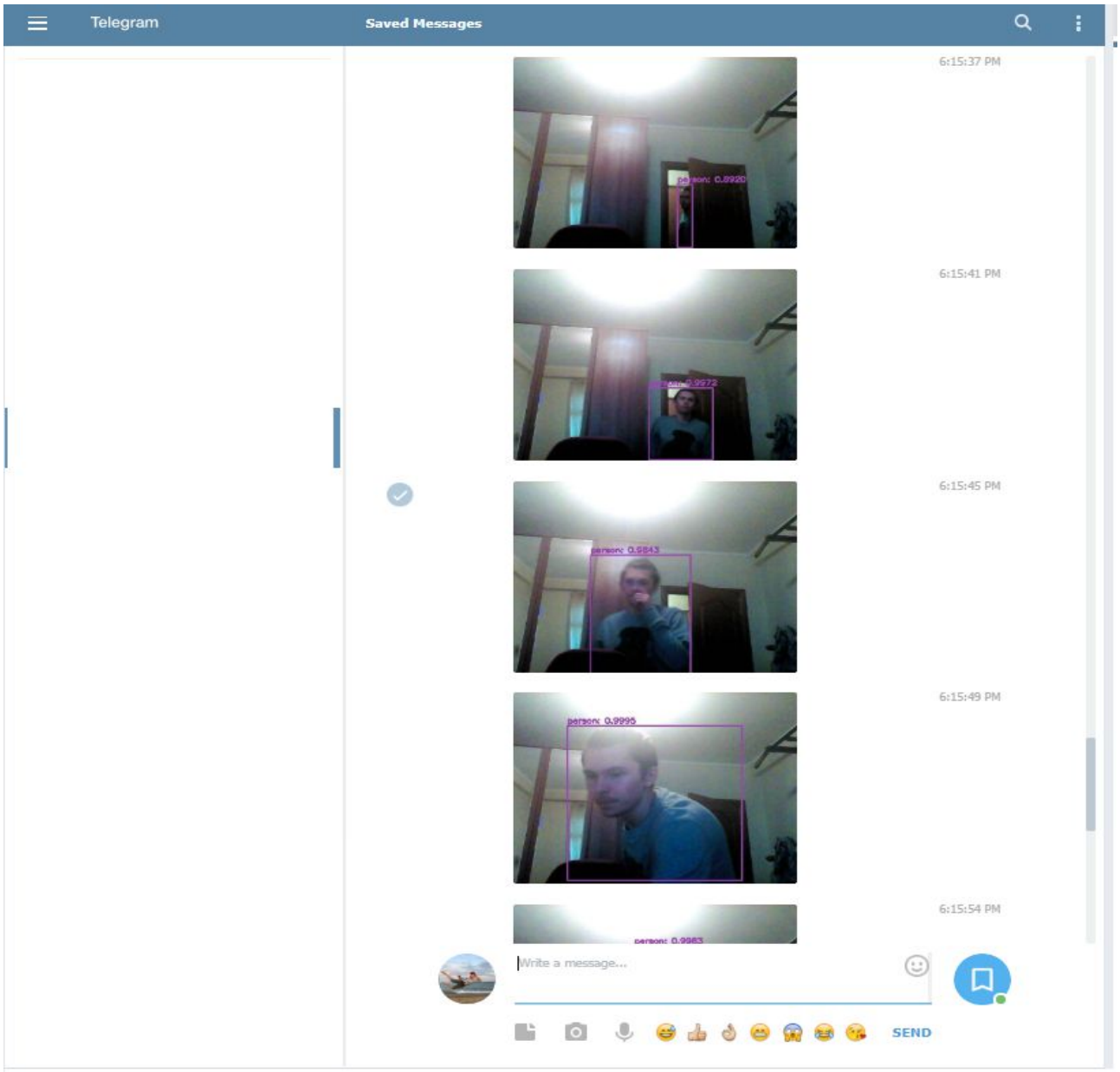

Рисунок 4.8 – Чат в який одразу надійшли фотографії

Висновки до розділу

Можливість обраної моделі CNN дає можливість навчити її розпізнавати об’єкти з великою точністю за рахунок transfer learning. Розроблена CNN ідеально підійшла для ідентифікації об’єктів, але під час тестувань, виникли проблеми з швидкістю і завантаженістю на процесор. Для цього, програму було модифіковано і результати помітно покращились. Для того, щоб показати роботу програми на реальному прикладі, було обрано концепцію “розумного” будинку. Щоб реалізувати систему відправки повідомлень, було обрано API

Telegram, яка на сьогодні дужа популярна. Тестування і результати програми виявились позитивними.

ВИСНОВКИ

В результаті виконання дипломного проєкту було розроблено програму ідентифікації рухомого об’єкту в відео-потоці і зображені. Створена система може бути використана в захисних системах, системах контролю і навігації, а також фіксування активності в різних областях.

Було проаналізовано методи та алгоритми обробки зображень, а також методи знаходження об’єктів на зображенні.

Також,були розглянуті інструменти та платформи для реалізації поставленої задачі. Продемонстрована ефективна робота бібліотеки OpenCV у сфері відстеження об’єктів.

Розроблено архітектуру програми і алгоритм роботи. Як мову програмування було обрано Python, а в якості системи відстеження обрано загорткову нейронну мережу.

Були проаналізовані різні моделі нейронних мереж, і обрано найефективнішу YOLOv3. В цілях вирішення поставленої задачі, а також покращення швидкості праці цієї моделі була створена своя модель з меншою кількістю класів для класифікації об’єктів. Також було досліджено і модифіковано роботу програми в цілях покращення швидкості і зменшення навантаження на комп’ютер. Для модифікації програми було використано більшість методів обробки зображень, які досліджувались в дипломному роєкті.

Як приклад застосування розробленої системи, було створену програму яка б заміняла пристрої “розумного” дому. Дана програма здатна фіксувати рух в відео-потоці і відправляти повідомлення про це власнику програми.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Методи знаходження спеціальних точок : URL : https://infopedia.su/17x9179.html (дата звернення 13.04.2020)
2. SIFT algorithm : URL : https://en.wikipedia.org/wiki/Scale-invariant_feature_transform (дата звернення 13.04.2020)
3. SURF algorithm : URL : https://en.wikipedia.org/wiki/Speeded_up_robust_features (дата звернення 13.04.2020)
4. ORB algorithm : URL : https://medium.com/data-breach/introduction-to-orb-oriented-fast-and-rotated-brief-4220e8ec40cf (дата звернення 13.04.2020)
5. Understanding of CNN : URL : https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148 (дата звернення 27.04.2020)
6. Viola-Jones method: https://medium.com/datadriveninvestor/understanding-and-implementing-the-viola-jones-image-classification-algorithm-85621f7fe20b (дата звернення 27.04.2020)
7. YOLOv3 official web-site : URL : https://pjreddie.com/darknet/yolo/ (дата звернення 27.04.2020)
8. OpenCV tutorials : URL : https://docs.opencv.org/master/d9/df8/tutorial_root.html (дата звернення 30.04.2020)
9. Telethon Documentation : URL : https://docs.telethon.dev/en/latest/ (дата звернення 7.05.2020)